TÓPICOS ESPECIAIS EM SISTEMAS DE INFORMAÇÃO

Minicursos do XVIII Simpósio Brasileiro em Sistemas de Informação (SBSI 2022)

Organizadores:

Rafael Dias Araújo (UFU)
Mariangela Setti (UTFPR)
Rita Cristina G. Berardi (UTFPR)
Alexandre R. Graeml (UTFPR)

DAINF
Escola de Gestão e Economia Dagee
UTFPR
CESI
SBC

Organizadores
Rafael Dias Araújo
Mariangela Gomes Setti
Rita Cristina G. Berardi
Alexandre R. Graeml

TÓPICOS ESPECIAIS EM SISTEMAS DE INFORMAÇÃO
Minicursos SBSI 2022

Sociedade Brasileira da Computação
Porto Alegre
2022

TÓPICOS ESPECIAIS EM SISTEMAS DE INFORMAÇÃO
Minicursos SBSI 2022

Sociedade Brasileira de Computação –SBC
CNPJ: 29.532.264/0001-78

Coordenação Geral
Rita Cristina G. Berardi
Alexandre R. Graeml

Coordenação do Comitê de Programa - Minicursos
Rafael Dias Araújo
Mariangela Gomes Setti

Edição dos Anais
Williamson Silva

Realização

Organização

Em cooperação

Editores

Comitê técnico

Comissão Especial de Sistemas de Informação da SBC (CESI)

Comitê de Programa – Minicursos

Alex Borges - UFJF
Andre Freire - UFLA
Andre Martinotto - USC
Carla Merkle Westphall - UFSC
Carlos Eduardo Santos Pires - UFCG
Claudia Cappelli - UFRJ
Daniel Notari - UCS
Daniela Barreiro Claro - UFBA
Edmundo Spoto - UFG
Eliomar Lima Federal - UFG
Emanuel Coutinho - UFC
Emilio Francesquini - UFABC
Fernanda Santos -UFU
Flávio Soares Corrêa da Silva - USP
Glauco Carneiro - UNIFACS
Heitor Costa - UFLA
Jorge Barbosa - UNISINOS
José Maria David - UFJF
Juliano Lopes de Oliveira - UFG
Leonardo Guerreiro Azevedo - IBM Research - Brazil
Leticia Peres - UFPR
Marcos Chaim - USP
Maria Istela Cagnin - UFMS
Mariangela Gomes Setti - UTFPR
Morganna Diniz - UNIRIO
Patricia Vilain - UFSC
Paulo Sérgio Santos - UNIRIO
Renata Araujo - UPM
Ricardo Choren - IME/RJ
Rodrigo Miani - UFU
Rodrigo Santos - UNIRIO
Scheila de Avila e Silva - UCS
Thiago Pirola Ribeiro - UFU
Victor Stroele Federal - UFJF

Dados Internacionais de Catalogação na Publicação

Simpósio Brasileiro de Sistemas de Informação (18.: 16-19 maio 2022: Curitiba, PR)
Tópicos especiais em sistemas de informação [recurso eletrônico] : minicursos do XVIII Simpósio Brasileiro de Sistemas de Informação (SBSI 2022) / organização: Rafael Dias Araújo [et al.] -- Porto Alegre: Sociedade Brasileira de Computação, 2022.
1 arquivo texto (50 f.): PDF; 3,7 MB.

Acesso via World Wide Web.
Inclui bibliografias.
eISBN: 978-65-87003-90-0

1. Tecnologia da informação - Brasil - Congressos e convenções. 2. Empresas novas. 3. Produtos novos. 4. Empreendedorismo. 5. Inovações tecnológicas. 6. Aprendizado profundo (Aprendizado de máquina). 7. Processamento de linguagem natural (Computação). I. Araújo, Rafael Dias [et al.]. II. Universidade Tecnológica Federal do Paraná. III. Sociedade Brasileira de Computação. IV. Título.

CDD: Ed. 23 – 004.6

Biblioteca Central do Câmpus Curitiba - UTFPR
Bibliotecária: Luiza Aquemi Matsumoto CRB-9/794

Prefácio Minicursos SBSI 2022

Este livro reúne trabalhos apresentados nos minicursos ministrados no XVIII Simpósio Brasileiro de Sistemas de Informação (SBSI 2022), organizado pela Universidade Tecnológica Federal do Paraná, no período de 16 a 19 de maio de 2022. Participam do SBSI, fórum nacional de debates da área de Sistemas de Informação (SI), estudantes e pesquisadores com apresentação de trabalhos científicos e discussão de temas contemporâneos relacionados à área. Esta edição, que aconteceu ainda durante a pandemia de COVID-19, foi realizada de forma híbrida (on-line e presencial).

Neste ano, foram submetidas 04 (quatro) propostas de minicursos válidas e 02 (duas) foram selecionadas, tal qual decidido em reunião da Comissão Especial de Sistemas de Informação. Tais propostas foram avaliadas por, no mínimo, três pesquisadores que fazem parte de um comitê composto por 34 professores pesquisadores.

Os dois minicursos contidos neste livro abordam tópicos de interesse da comunidade de Sistemas de Informação e correlatos ao tema da 18ª edição do evento, cujo tema foi “Sistemas de Informação para um Mundo mais Humano”. O primeiro capítulo, intitulado “*Como Organizar uma Hackathon Corporativa?*”, aborda o processo realizado para organizar e conduzir hackathons corporativas. O segundo capítulo, intitulado “*Deep Learning para Processamento de Linguagem Natural*”, discute técnicas de processamento de linguagem natural, dentre elas, algoritmos de aprendizado profundo.

Esperamos que este livro auxilie estudantes, pesquisadores e profissionais da área de Sistemas de Informação na construção do conhecimento em temas específicos relacionados ao que foi aqui apresentado.

Rafael Araújo (UFU) e Mariangela Gomes Setti (UTFPR)
Coordenadores da Trilha de Minicursos do SBSI 2022
Curitiba/PR, Maio de 2022.

Coordenadores dos Minicursos do SBSI 2022

Rafael Dias Araújo é Bacharel, Mestre e Doutor em Ciência da Computação pela Universidade Federal de Uberlândia. Realizou um estágio na área de desenvolvimento de software na France Telecom durante um período de intercâmbio acadêmico entre o Institut National des Sciences Appliquées de Lyon (INSA, França) e a UFU. Realizou um período de estágio de doutorado sanduíche durante um ano no laboratório de Sistemas Web Adaptativos e Personalizados (PAWS) na School of Information Sciences (iSchool) da Universidade de Pittsburgh (USA). Atualmente é professor adjunto da Faculdade de Computação (FACOM/UFU) - Campus Monte Carmelo e professor do Programa Pós-Graduação em Ciência da Computação (PPGCO/UFU), membro do Comitê Gestor (2018-2020) da Comissão Especial da Informática na Educação (CEIE) e do Comitê Gestor (2020-2021) da Comissão Especial de Sistemas de Informação (CESI) da Sociedade Brasileira de Computação (SBC) e editor-chefe da Revista Brasileira de Informática na Educação (RBIE). É pesquisador nas áreas de Recomendação e Personalização de Conteúdo, Computação Ubíqua, Interação Humano-Computador, Sistemas Web e Multimídia Interativos, Informática na Educação e Sistemas de Informação.

Mariangela Gomes Setti é professora titular da Universidade Tecnológica Federal do Paraná, com atuação nos cursos de bacharelado em Sistemas de Informação. Atualmente é Coordenadora do Curso de BSI, pesquisa a Aprendizagem de Introdução à Programação (Algoritmos) e a participação das mulheres na área de Computação. Doutora em Educação - linha de Educação Matemática, pela UFPR (2009). Possui graduação em Ciência da Computação pela Universidade Estadual de Maringá (1993) e mestrado em Engenharia Elétrica e Informática Industrial pela Universidade Tecnológica Federal do Paraná (1999).

Organizadores Gerais do SBSI 2022

Rita Cristina Galarraga Berardi é professora adjunta na Universidade Tecnológica Federal do Paraná, Campus Curitiba. Doutora em Ciência da Computação pela Pontifícia Universidade Católica do Rio de Janeiro, tendo realizado parte da pesquisa na Universität Koblenz-Landau, Instituto West de Web Semântica na Alemanha. Completou o mestrado em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul, em parceria com Hewlett Packard (HP). Graduada em Ciência da Computação pela Universidade Federal de Pelotas. As áreas de pesquisa que mais interessam estão relacionadas a Banco de Dados, Qualidade dados, Linked data (ou dados conectados), Web Semântica e Ontologias, todas essas áreas associadas a problemas reais de aplicação prática, como em cidades inteligentes. Na Extensão, é cofundadora do Projeto TIChers (@tichers_utfpr instagram) da UTFPR que visa trabalhar com professoras da Educação Básica apresentando conceitos de computação para atrair mais mulheres para a computação. A Profa. Rita Cristina Galarraga Berardi tem atuação marcante nos eventos da SBC, tendo participado dos Congressos da Sociedade Brasileira nos últimos 3 anos em eventos satélites como Workshop em Ensino de Computação (WEI), Workshop de Informática na Escola (WIE), Simpósio de Informática na Saúde (SBCAS), Simpósio de Informática na Educação (SBIE). Foi coordenadora do Workshop de Teses e Dissertações em Ontologias (WTDO) em 2019, em 2020 foi Coordenadora da Sessão “Sua pesquisa em um vídeo” do ISLA (Information Systems in Latin America). Em 2021 foi a Coordenadora do Comitê de Programa do ONTOBRAS 2021, assim como é Coordenadora Geral do XV Women in Information Technology – WIT 2021, evento satélite do CSBC 2021.

Alexandre Reis Graeml é professor titular da Universidade Tecnológica Federal do Paraná, com atuação nos cursos de bacharelado em Sistemas de Informação e nos programas de pós-graduação em Computação Aplicada e Administração. Sua experiência em Ensino e pesquisa em Sistemas de Informação envolve duas passagens pela Universidade da Califórnia - Berkeley, como research associate (2002) e visiting scholar (2018/2019) além da atuação como professor no mestrado em Sistemas de Informação da Ecole Supérieure d’Ingénieurs (Esigelec) na França (desde 2010). Atua ainda como consultor do National Center for Academic Accreditation and Assessment, da Arábia Saudita, avaliando programas das áreas de ciência da computação, engenharia de software e sistemas de informação daquele país (desde 2017). Seus interesses de pesquisa envolvem o impacto da TI nas organizações e na sociedade, adoção, aceitação e apropriação de sistemas de informação, o fluxo de informação ao longo de cadeias de suprimento, gestão do conhecimento e inteligência coletiva. É atualmente presidente do capítulo latino-americano da Association for Information Systems (AIS). Embora atue mais diretamente na

comunidade acadêmico/científica de Administração da Informação, tem um relacionamento de longa data com o SBSI, tendo organizado o III SBSI, em 2006, juntamente com o Prof. Gustavo Lugo, quando ambos eram professores da Universidade Positivo. Em edições posteriores, foi convidado para dar palestras ou participar de mesas redondas em outras duas ocasiões. Em 2014, em Londrina, participou de painel do XIII SBSI, com Renata Araújo, Célia Ralha e Alexandre Cidral debatendo os “Desafios e oportunidades para o futuro” do SBSI, passados dez anos desde a primeira edição do evento. Em 2020, no XVII SBSI, proferiu a Master Class “Teoria e Método: Sistemas de Informação no Estudo de Relações Sociais”.

Sumário

Capítulo 1

Como Organizar uma Hackathon Corporativa?

George Valença, Rodrigo Santos

Abstract

Public and private organizations have adopted open innovation initiatives to establish a sustainable digital transformation process, increasing efficiency and offering value to society. In this context, corporate hackathons often appear as tools to create innovative solutions and develop specific skills, which involves integrating the tripod of the Information Systems area (people, processes/organizations, and technologies). This chapter presents a process for conducting corporate hackathons, allowing participants to adopt this tool to improve their organizations' work processes, products, and services from the perspective of innovation. The chapter is divided into two parts. The first one presents basic concepts about hackathons, such as origin, types, and examples. The second part presents an example of a dynamic of organizing a corporate hackathon, which is discussed through the Problem-Based Learning approach. Finally, the chapter points out some challenges and lessons learned.

Resumo

Organizações públicas e privadas têm adotado iniciativas de inovação aberta para estabelecer um processo de transformação digital sustentável, aumentando a eficiência e oferecendo valor à sociedade. Nesse contexto, as hackathons corporativos surgem muitas vezes como ferramentas para criar soluções inovadoras e desenvolver habilidades específicas, o que envolve integrar o tripé da área de Sistemas de Informação (pessoas, processos/organizações e tecnologias). Este capítulo apresenta um processo para a realização de hackathons corporativos, permitindo que os participantes adotem essa ferramenta para melhorar os processos de trabalho, produtos e serviços de suas organizações na perspectiva da inovação. O capítulo está dividido em duas partes. A primeira apresenta conceitos básicos sobre hackathons, como origem, tipos e exemplos. A segunda parte apresenta um exemplo de dinâmica de organização de um hackathon corporativo, discutido através da abordagem de

Aprendizagem Baseada em Problema. Por fim, o capítulo aponta alguns desafios e lições aprendidas em suas considerações finais.

1.1. Introdução

Nos últimos anos, organizações públicas e privadas têm adotado iniciativas de inovação aberta para abraçar o paradigma de transformação digital, de forma a aumentar sua eficiência e oferecer resultados de valor para o cliente (cidadão, órgãos parceiros etc.) (VALENÇA *et al*., 2019). Neste contexto, as *hackathons* corporativas se tornaram frequentes, sendo entendidas como eventos em que uma organização reúne diferentes participantes (e.g., profissionais, pesquisadores, estudantes) em equipes para colaborar intensamente na criação de soluções inovadoras e desenvolver habilidades específicas (VALENÇA *et al*., 2020a). Consequentemente, tais eventos podem ser utilizados como ferramentas para analisar os desafios organizacionais de forma criativa, com a identificação de oportunidades para inovar seus produtos e serviços a partir de um olhar tanto interno (funcionários) quanto externo (parceiros ou terceiros, como estudantes ou *startups*). Essas abordagens colaborativas buscam desenvolver soluções (geralmente de TI, de protótipos a MVP – *Minimum Viable Products*) a partir da visão de múltiplos participantes, que se dedicam à resolução de um problema por um período de tempo restrito, que varia de 24 horas a uma semana (PE-THAN *et al*., 2019).

No âmbito da área de Sistemas de Informação (SI), que tem como princípio a busca por soluções para problemas do mundo real, da sociedade e das organizações, por meio de novas tecnologias, as *hackathons* são um tema de pesquisa e prática de grande relevância. Considera-se o fato de que a pesquisa em SI foca em investigar, de maneira integrada, problemas e soluções com base no tripé formado por pessoas, processos/organizações e tecnologias (ANTONIO *et al*., 2021; STEGLICH *et al*., 2021). Uma vez que as tecnologias trazem mudanças que afetam o ambiente onde operam e as pessoas que vivem neste ambiente, mesmo que não as utilizem, a complexidade em se projetar e usar SI tem requerido melhor compreensão de como a colaboração intensa favorece a criação de soluções inovadoras para produtos e serviços e como desenvolver habilidades específicas neste contexto, como nas *hackathons*. Isto, inclusive, é apontado em alguns desafios presentes nos Grandes Desafios de Pesquisa em SI no Brasil (GranDSI-BR), como "*Information Systems and the Open World Challenges*", "*Methodologies and Technologies for Citizen Participation*" e "*Systemic and Socially Aware Perspective for Information Systems*", o que reforça a importância das *hackathons* corporativas (BOSCARIOLI *et al*., 2017).

Adicionalmente, a partir do tema "*Sistemas de Informação para um Mundo mais Humano*" e pelo fato de SI lidar com situações inesperadas e promover uma gestão mais eficiente e transformadora nas mais diversas áreas, como saúde, educação, logística, agronegócio, finanças, gestão pública, causas sociais, dentre outras (GRACIANO NETO *et al*., 2020), este capítulo vem então a contribuir com conceitos e aplicações que pesquisadores e profissionais devem ter em mente em um cenário que envolve novos tipos e interações com/de SI. Busca-se demonstrar, de maneira simples e didática, a importância de se conduzir *hackathons* corporativas, permitindo a adoção de um processo claro para explorar essa ferramenta a fim de melhorar os processos de trabalho,

produtos e serviços de suas organizações, sob a ótica de inovação, inclusive promovendo uma dinâmica com os participantes.

Dessa forma, o objetivo principal deste capítulo é apresentar um macroprocesso para condução de *hackathons* corporativas, permitindo que os interessados as adotem para melhorar os processos de trabalho, produtos e serviços de suas organizações, que são resultados esperados para esta ferramenta de inovação aberta (NOLTE *et al*., 2018). As três etapas para organização destes eventos são apresentadas de forma a permitir a sua compreensão prática, à luz de um trabalho colaborativo entre os interessados acerca de problemas reais de suas organizações. No cenário da pandemia da COVID-19, destaca-se que as *hackathons* corporativas podem ocorrer tanto em uma sala virtual (e.g., uso do Google Meet e Google Jam Board entre os participantes) como em uma sala presencial (e.g., espaço de *co-working* ou laboratório). Vale ressaltar ainda que o escopo deste capítulo é introdutório e voltado para pesquisadores, professores e estudantes, bem como profissionais da indústria interessados no assunto.

Além desta seção de introdução, este capítulo está organizado da seguinte forma: a Seção 1.2 apresenta conceitos básicos acerca de *hackathons*, incluindo definições, características, benefícios, dificuldades e desafios; a Seção 1.3 apresenta o macroprocesso proposto para condução de uma *hackathon* corporativa, detalhando as suas fases, duração, participantes e localização, além de listar boas práticas; a Seção 1.4 apresenta dois cenários de aplicação do macroprocesso proposto a partir de casos reais, a título de exemplificação; por fim, a Seção 1.5 conclui o capítulo com algumas considerações, bem como implicações para a teoria e prática em SI.

1.2. Conceitos Básicos

Nos últimos 20 anos, o termo *hackathon* mudou bastante em relação ao seu enfoque inicial, partindo de um viés simples de eventos que reúnem desenvolvedores para codificar aplicações em direção a eventos com estratégias robustas. Atualmente, *hackathons* são entendidas como eventos com prazo determinado em que as pessoas se reúnem para criar e potencialmente lançar uma nova solução para um problema específico, construída com base em tecnologia nova ou existente (ROSELL *et al*. 2014). Empresas de grande porte de Tecnologia da Informação (TI), como Google e Facebook, ou mesmo pequeno porte, como In Loco, localizada no Porto Digital da cidade do Recife, realizam *hackathons* como mecanismo para explorar e concretizar a chamada inovação aberta (CHESBROUGH, 2003). Independente do caso, a inovação acontece em qualquer lugar, inclusive além das fronteiras da organização, a fim de assegurar a sua sobrevivência por não conseguir construir ou se apropriar de todos os aspectos de seu modelo de negócio (GOLDMAN & GABRIEL, 2005).

No contexto de inovação aberta, é importante destacar que as inovações não originam de uma única organização como costumava ocorrer, mas envolvem também "co-inovação" a partir de diferentes atores da indústria de software. É sabido que, há algum tempo, as inovações também compreendem processos de "co-evolução", que são possíveis apenas a partir da interação de organizações, com foco em colaboração, a fim de dar suporte a novos produtos, satisfazer as necessidades dos clientes e incorporar novas rodadas de inovação (ARNDT & DIBBERN, 2006). Por exemplo, as *hackathons*, do ponto de vista das organizações que produzem produtos e serviços de software,

visam lidar com o desenvolvimento de software de larga escala, que é complicado, custoso, lento e imprevisível (BOSCH & BOSCH-SIJTSEMA, 2010). Lidar, de forma integrada, com as questões sociais e econômicas junto aos aspectos técnicos do desenvolvimento de software se torna indispensável (BOEHM, 2006; COSTA *et al*. 2021), o que requer profissionais com a habilidade de abstrair a complexidade do sistema como um todo.

De modo geral, esses eventos podem atender aos mais diversos objetivos, tais como incrementar o portfólio da empresa, contratar novos funcionários, integrar setores da sociedade, solucionar problemas sociais emergenciais, promover diversidade e inclusão etc. Eventos como *hackathons* podem agregar produtos, testar plataformas ou testar novos recursos disponibilizados pela organização. Um exemplo: a *hackathon* da Uber[1] traz na sua descrição a importância de oferecer uma API (*Application Programming Interface*) para que participantes criem inovações dentro de sua plataforma. Vale ressaltar que isto se deve ao fato de que a plataforma, neste caso, é vista como um sistema que representa uma combinação de software, hardware e "peopleware", constituída sobre um ambiente comum, cujas equipes têm trabalhado geograficamente dispersas e envolvem participantes externos à organização. Tal estratégia resulta do fato de que as organizações têm sofrido pressão do mercado para a abertura das plataformas e o envolvimento de atores externos (SANTOS *et al*., 2020). A premissa é que, independente de quão espertos, criativos e inovadores sejam seus funcionários e demais envolvidos internos, sempre existirão outros profissionais de nível semelhante ou superior fora de suas fronteiras que podem contribuir para manter sustentável a dinâmica de oferta-demanda (BURMANN & MAURO, 2011).

De acordo com Porras *et al*. (2018), hackathons são caracterizadas por atingirem objetivos específicos no contexto em que estão inseridas, trazendo visões distintas e agregando diferentes grupos acerca de um mesmo objetivo. Os seguintes tipos de *hackathons* têm sido discutidos, como mostra a Figura 1.1: *educacionais*, *cívicos* e *corporativos*. Como exemplo de *hackathons* educacionais, pode-se citar o evento Edu[2], organizado pelo SENAC de Pernambuco, que tem por objetivo estimular ideias ligadas aos desafios previamente propostos aos participantes na temática educacional, como palestras, *meetups* etc. Por sua vez, como exemplo de *hackathons* cívicas, pode-se mencionar o evento OpenGovData[3], focado em soluções centradas no cidadão (e.g., aplicações móveis/web, *bots* etc.) utilizando dados abertos. Por fim, *hackathons corporativas* são utilizadas como forma de otimizar a inovação em produtos, tornando-os comercializáveis ao final do processo, visando não só agregar inovação ao negócio das organizações, mas também divulgá-lo para atrair novos talentos para fortalecer a sua missão e produtividade.

Mais especificamente, *hackathons* corporativas atuam como instrumentos de inovação e que promovem colaboração na criação de ideias, protótipos ou planos de negócio (PE-THAN *et al*., 2019). A partir da sua finalidade, podem ser divididas em

[1] www.uberhackathon.devpost.com

[2] www.pe.senac.br/cte/hackathon

[3] www.innovate.mygov.in/opengovdatahack2019

eventos *internos* ou *externos*. *Hackathons internas* têm o objetivo principal de catalisar a inovação das organizações que as promovem, levando os seus funcionários a um ambiente diferente do comum para que inovem para além de sua rotina, trazendo novos desafios e ideias para o seu contexto de trabalho (HERALA *et al*., 2019). Por outro lado, as *hackathons externas* reúnem tanto funcionários da organização como atores externos, que trazem consigo experiências diversas, interferindo diretamente na evolução dos seus produtos internos (CHESBROUGH, 2003). Em ambos os casos, a organização busca desenvolver mecanismos para acelerar a sua produção por meio da inovação aberta ao promover a colaboração com novos parceiros, compartilhar custos e incorporar novos serviços e funcionalidades em seus produtos à luz do mercado, caso seja atestado o seu sucesso (FONTÃO *et al*., 2021).

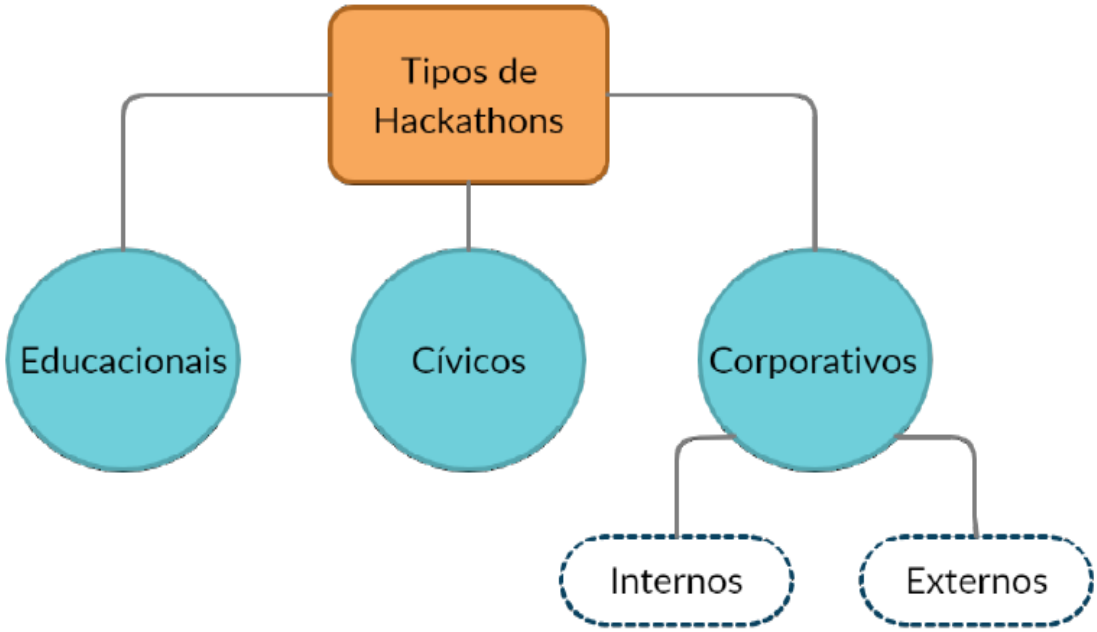

Figura 1.1. Tipos de *hackathons*.

As *hackathons* corporativas estimulam, nesse sentido, a filosofia de "abertura" das organizações, como mecanismo estratégico para inovação de produtos, processos ou serviços. Dois requisitos são atendidos nestes eventos em termos da potencial novidade: gerar ganho social, por permitir que mais bens e serviços sejam entregues à sociedade; e gerar retorno para a organização (financeiro e outros), por ser consumida por ela (SANTOS, 2010). Em alguns cenários, tais eventos ocorrem com base em uma infraestrutura digital auto-organizável que cria um ambiente digital para as organizações (e seus atores) conectadas em rede, dando suporte à cooperação, ao compartilhamento de conhecimento e ao desenvolvimento de tecnologias adaptativas e abertas, além de constituírem espaços ricos de conhecimento, o que forma e fomenta os chamados ecossistemas digitais (BOLEY & CHANG, 2007). Isto contribui diretamente para transformar o processo da *inovação fechada*, i.e., cada organização promove um processo de inovação internamente, para se criar um processo de *inovação aberta*, no qual existem intermediários para agir como mediadores ou avaliadores das diferentes contribuições desenvolvidas sobre a sua plataforma e gerar relatórios ou encaminhamentos (IANSITI & LEVIEN, 2004), como ilustra a Figura 1.2.

Finalmente, o esforço dos atores externos em contribuir para organizações que promovem *hackathons* pode transformar a inovação em algo positivo para ambos, ao mesmo tempo em que contribui para as organizações estenderem ou melhorarem seus negócios e aumentarem o número de envolvidos, inclusive clientes e usuários, o que corrobora a visão de um ecossistema, conforme mencionado em (HANSSEN, 2012). A proximidade entre estes três grupos se torna fundamental para o sucesso e evolução de qualquer empreendimento. Entretanto, vale destacar que manter uma organização competitiva em um mercado altamente globalizado e interdependente como hoje é completamente diferente de dominar uma solução ou tecnologia, pois requer lidar de forma integrada com o tripé de SI, i.e., pessoas, processos/organizações e tecnologias.

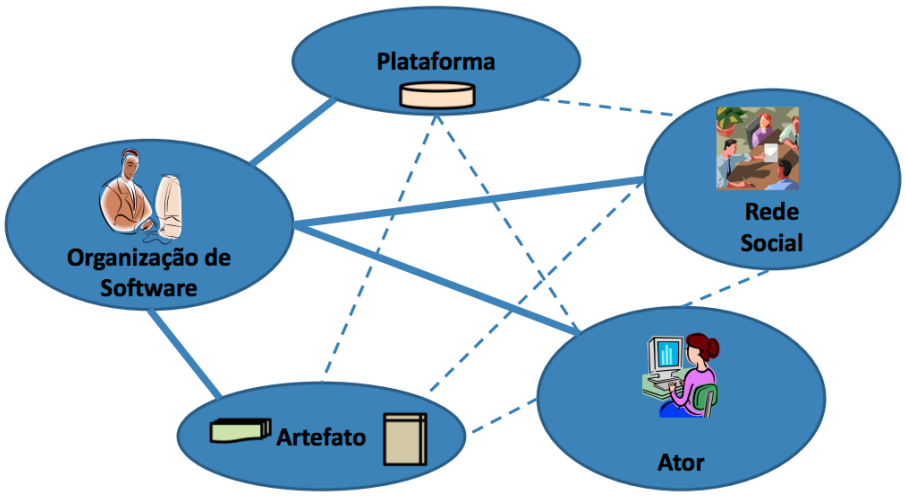

Figura 1.2. Fontes de inovação fechada (tradicional), cujas linhas pontilhadas são relações que não estão estabelecidas claramente, dado que a organização centraliza as relações, que deveriam ser fomentadas em direção à inovação aberta, por exemplo, apoiando e sendo apoiadas por *hackathons* corporativas. Fonte: (SANTOS, 2013; 2016).

Nas próximas seções, conceitos e elementos relacionados com *hackathons* corporativas são detalhados. O objetivo é introduzir de maneira mais simples o que é importante conhecer quando se pensa em organizar e conduzir eventos como estes.

1.3. Organizando uma Hackathon Corporativa

Pode-se dividir a organização de uma *hackathon* em três fases: ***pré-hackathon***, ***hackathon*** e ***pós-hackathon***. Juntas, estas fases compõem um macroprocesso de dez atividades, conforme mostra a Figura 1.3. Todas as atividades marcadas com (*) são aquelas cuja realização se mostra essencial para a organização de uma *hackathon*. As demais atividades são consideradas opcionais ou acontecem em situações ocasionais. Além disso, as atividades são sequenciais, com exceção da *mentoria técnica* (A8), que simboliza o suporte aos participantes da *hackathon*, que pode ser feita paralelamente às demais atividades.

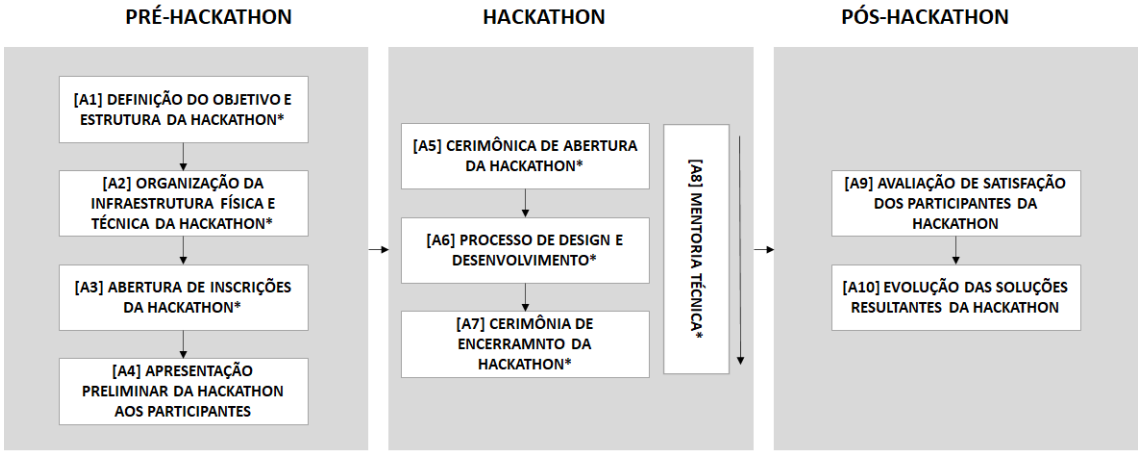

Figura 1.3. Processo em três fases e dez atividades para *hackathons* corporativas. Fonte: Adaptado de (VALENÇA *et al*., 2020a).

1.3.1. Fases

A ***pré-hackathon*** (primeira fase da Figura 1.3) busca definir os principais aspectos do evento, tais como (i) se o evento será interno, externo ou híbrido; (ii) se o evento irá durar alguns dias ou uma semana; e (iii) se o local onde o evento ocorrerá corresponde ao *campus* da universidade ou ao departamento da própria organização. Os detalhes destas escolhas são discutidos na seção de objetivos do evento.

Essa fase se inicia com a *definição do objetivo e estrutura da hackathon* (A1), em que os organizadores definem quais serão os desafios atrelados ao evento, como representar problemas relacionados a determinadas áreas (e.g., bioinformática e genômica). Estes desafios não necessariamente são interessantes apenas para os participantes. Além disso, eles podem não ser passíveis de exploração em um curto intervalo de tempo. Por exemplo, os organizadores podem definir casos de uso a explorar em metade de um dia de trabalho. Entretanto, esta prática não é comum em *hackathons* pelo fato de restringir o escopo de inovação.

Após a execução da atividade A1, a *organização da infraestrutura física e técnica da hackathon* (A2) é colocada em prática. Na atividade A2, os organizadores procuram possíveis locais, ou seja, devem tomar a decisão sobre a modalidade do evento, quer seja em um local interno da organização ou fora dela (*campus* de universidades, espaço de inovação etc.). A meta principal está em trazer benefícios tanto aos participantes como à organização. Por exemplo: os organizadores podem procurar um local que promova o máximo de diversidade regional, além de acomodar os diferentes participantes com os mais variados perfis. Outro foco importante nesta etapa consiste na preparação da infraestrutura técnica do local, instalando software, acesso à rede (Internet) etc. Por exemplo: pode ser preciso instalar um software para realizar o evento, ou criar contas para os usuários.

A partir da finalização da atividade A2, é inicializada a *abertura de inscrições da hackathon* (A3). Esta abertura ocorre geralmente direcionada a quem vai explorar a tecnologia exposta no evento. No entanto, caso não sejam convidados a atuar no evento como organizadores, parceiros (como fornecedores da tecnologia) poderão realizar esta etapa de registro – até indicando possíveis desafios a serem resolvidos no evento. Esta

atividade pode ser organizada com maior detalhamento, como a criação de uma agenda que englobe um plano de marketing e abertura de inscrição de participantes. Geralmente, são usados *sites* ou formulários web para que participantes internos e/ou externos se registrem na *hackathon*. Por exemplo, uma ferramenta para que os funcionários da organização indiquem os seus projetos, procurem membros interessados naquele projeto e criem uma equipe para o evento. Neste processo de inscrição, pode ser necessário que o candidato descreva quais são as suas principais habilidades e ideias para o evento, como também escolha qual tema ele prefere explorar no evento.

Após a finalização da inscrição dos possíveis candidatos, os organizadores devem analisar os inscritos, podendo até criar os times que irão realizar o evento. A seleção dos candidatos é baseada em habilidades, objetivos e opiniões. Por exemplo, a organização pode agrupar *stakeholders* internos em times temáticos (i.e., profissionais de *marketing*, engenheiros e arquitetos). Outra opção seria reuni-los de modo a maximizar a diversidade de opiniões e habilidades, ou formar times baseados nos interesses comuns com outros participantes – esta abordagem é facilitada quando os organizadores conectam os grupos utilizando redes sociais, como a ferramenta Slack[4].

Com a definição dos participantes, os organizadores podem realizar uma atividade opcional, chamada da *apresentação preliminar da hackathon aos participantes* (A4). Essas apresentações podem ocorrer de forma on-line (via Internet), ou em eventos presenciais, entre uma e quatro semanas antes do evento ocorrer. O principal ponto é descrever o tema ou os temas principais do evento, como a adição da descrição de documentação, artefatos e ferramentas presentes na *hackathon*. Por exemplo: compartilhar a SDK (*Software Development Kit*) com os participantes para que eles se familiarizem antes do evento.

As organizações sempre iniciam a ***hackathon*** (segunda fase da Figura 1.3) com a atividade de *cerimônia de abertura* (A5), que consiste geralmente no detalhamento de como será a condução do evento. Este detalhamento envolve não apenas a divulgação do tema e dos desafios a serem enfrentados pelos participantes, mas também as regras, instruções e expectativas de como será conduzido o processo da *hackathon*. Nesta atividade, também é conduzido todo o processo de permissões e arranjo da infraestrutura para a preparação do participante (e.g., autorização para acessar os ambientes de desenvolvimento, versionamento de código etc.). Isto ocorre sempre nas horas iniciais do evento.

Ao iniciar o evento, os participantes têm os primeiros contatos com as tecnologias utilizadas e adotadas pelos organizadores do evento por meio de uma "apresentação técnica", que pode envolver a apresentação da API que será utilizada no evento, por exemplo. É possível ainda que realizem tutorias e treinamento nas tecnologias oferecidas para os times do evento. Estas atividades podem ocorrer durante todo o evento, como uma *mentoria técnica* (A8). Nesta atividade, os times podem ser formados (caso não tenham sido definidos previamente) para que então iniciem o *processo de design e desenvolvimento* (A6). Neste momento, os membros compartilham as suas ideias (*brainstorming*) e definem quais são as estratégias a seguir durante o evento. Os organizadores podem guiar essa discussão inicial para direcionar as equipes

[4] https://slack.com/intl/pt-br/

a entender quais são os valores-chave da organização a fim de que proponham soluções alinhadas com estes valores.

Visando entender melhor o contexto dos problemas, os times podem realizar uma coleta de dados (e.g., entrevistas com potenciais clientes). Com os resultados desta coleta, eles podem definir as soluções que mais se adéquem a este contexto, atingindo os desafios apresentados. Por exemplo: os times podem realizar uma apresentação inicial de 10 minutos descrevendo o que pretendem construir e qual é o planejamento. Com base nos *feedbacks* dos organizadores, podem então definir um conceito de *design* alinhado às expectativas do evento exatamente antes de começarem a codificar. Com a solução inicial definida, os times passam à prototipagem e/ou codificação. Exemplos de resultados são um novo conceito, plano ou protótipo da solução.

O formato de solução mais frequente nos eventos é a produção de protótipos apoiados pelo processo de *Design Thinking* (KIMBELL, 2015). Durante esta atividade, é fundamental manter as tutorias e assistência técnica para as tecnologias e infraestrutura em utilização. Outro aspecto importante é garantir a motivação dos participantes, o que pode ocorrer inclusive com atividades energizantes e dinâmicas para descontraí-los, bem como com visitas de chefias da organização, que podem oferecer um suporte adicional.

Com os protótipos desenvolvidos, é iniciada a *cerimônia de encerramento* (A7). Nela, ocorre a apresentação dos resultados pelos times, que são julgados por uma banca examinadora. Esta etapa pode variar de acordo com a seleção dos times, composição do júri e processo de votação. Por exemplo: as equipes podem ter 10 minutos para apresentar o *pitch* de suas soluções a executivos da organização. O júri pode ser formado por funcionários da organização, atores externos (e.g., clientes, parceiros) e/ou comissão organizadora (e.g., mentores), mas também pode ser composto por uma combinação desses grupos de avaliadores.

Cabe ao júri analisar as apresentações e o trabalho desenvolvido, fornecendo *feedback* e votando nas soluções mais interessantes. Os *feedbacks* vão de ideias para evoluir a solução a sugestões sobre como comercializá-la. Além disso, é uma oportunidade para os times serem convidados a participarem de processos de incubação, o que pode ocorrer por meio de um *workshop* de incubação. Por fim, os júris executam o processo de votação, que pode ocorrer a partir de uma plataforma on-line e anônima. Ao final, as melhores soluções são premiadas.

Na ***pós-hackathon*** (terceira e última fase da Figura 1.3), os organizadores devem proceder com a *avaliação da satisfação dos participantes da hackathon* (A9). Ao ouvir os participantes, é possível entender como melhorar a preparação e condução de próximos eventos. Por exemplo: realização de uma reunião de coleta de opinião on-line acerca da *hackathon*. Na sequência, é possível que haja a *evolução das soluções resultantes do evento* (A10), quando os times podem continuar o desenvolvimento da solução proposta (e.g., adicionar novas funcionalidades) com base no *feedback* recebido pelos jurados. Em particular, a organização pode dar suporte à evolução da solução, com uma espécie de incubação ou patrocínio. Por fim, os participantes podem atender sessões de discussão de lições aprendidas para analisar os resultados do evento (e.g., ideias, protótipos etc.) e estabelecer quais projetos serão continuados.

1.3.2. Duração

Uma *hackathon* corporativa pode durar de um dia a uma semana (cinco dias úteis). A duração mais frequente tende a ser de um a três dias. Essa duração pode afetar diretamente os resultados obtidos, já que o tempo disponibilizado envolve o quanto as equipes poderão se aprofundar no entendimento do problema investigado e na proposta/desenvolvimento de soluções. Uma recomendação importante é que eventos não sejam muito curtos a fim de dar aos participantes uma janela de tempo de trabalho ideal para inovação. Por sua vez, eventos longos podem retirar o sentimento de tensão positiva para a produção de resultados, prejudicando a concentração das equipes, que acabam se distraindo mais.

Assim, a definição do prazo de duração de uma *hackathon* deve estar em conformidade com os seus objetivos. Por exemplo, um período de tempo restrito, de um a dois dias, pode permitir o entendimento dos problemas e a colaboração entre os grupos. Se o objetivo for promover/estender uma tecnologia, cabe expandir a duração em um dia, de maneira a permitir a realização de um treinamento, interações com especialistas nos temas e prototipagem iterativa das soluções.

1.3.3. Participantes e Localização

O perfil dos participantes envolvidos no evento pode variar bastante. Há três composições principais: (i) com *participantes internos* (funcionários), (ii) com *participantes externos* (estudantes, pesquisadores etc.) e (iii) *híbridas* (participantes internos e externos). Em geral, *hackathons* tendem a envolver apenas participantes internos por fatores como: sigilo de informações (e.g., dados importantes da empresa precisam ser disponibilizados para o público externo gerar soluções), conveniência (e.g., não é preciso realizar uma seleção) e custos (e.g., muitas vezes, participantes externos encontram motivação em prêmios, o que pode onerar o evento). Caso haja participantes externos, os funcionários da organização atuam com outros profissionais, estudantes e pesquisadores engajados com tecnologias ou que atuam no mesmo domínio/setor.

A maioria das organizações realiza *hackathons* nas suas próprias instalações. É possível também que o evento seja conduzido em um ambiente tido como mais estimulante, que permita maior imersão e fomente a criatividade dos participantes, como em um hotel ou no *campus* de uma universidade. Por fim, é possível que uma *hackathon* adote uma abordagem híbrida de realização, podendo, por exemplo, ser iniciada na universidade e, após passar pelo processo de ideação e prototipação, os participantes apresentam as suas demonstrações na organização em si.

É preciso ressaltar que o evento pode ser conduzido inteiramente de forma remota (on-line). Para isso, são sugeridas ferramentas, não somente repositórios de informações, como Google Drive[5], Dropbox[6] ou GitHub[7], mas principalmente aquelas que consigam promover a interação apesar da distância entre os participantes, como

[5] https://www.google.com/intl/pt-BR/drive/

[6] https://www.dropbox.com/pt_BR/

[7] https://github.com/

Miro[8] e Google Jam Board[9]. Para comunicação regular entre o time, Discord[10], Slack e Google Meet[11] são boas opções. Na Figura 1.4, é apresentada uma estrutura que condensa os principais elementos de uma *hackathon* em termos de planejamento, ou seja, as suas fases, possível duração, formato de participação e possível localização.

1.3.4. Boas Práticas

Hackathons são eventos que podem estimular o processo de inovação. Assim, cabe às organizações (1) **envolver o máximo de funcionários possível, garantindo que eles pertençam a diferentes áreas ou departamentos**, uma vez que as suas visões complementares ampliam o entendimento dos desafios tratados pelo evento e a geração de ideias de solução. Além disso, para potencializar esse processo, cabe (2) **primar por inovação aberta**, com convite ao público externo (e.g., potenciais usuários e parceiros), ainda que haja desafios com relação à privacidade e segurança da informação no âmbito da organização (VALENÇA *et al.*, 2020b). Em paralelo, para além das soluções que podem derivar de uma *hackathon*, é preciso que as organizações reconheçam o potencial deste tipo de evento para desenvolver a competência de inovação, com habilidades como raciocínio criativo, comunicação simples e análise qualitativa sendo exigidas de seus participantes (e.g., funcionários da organização).

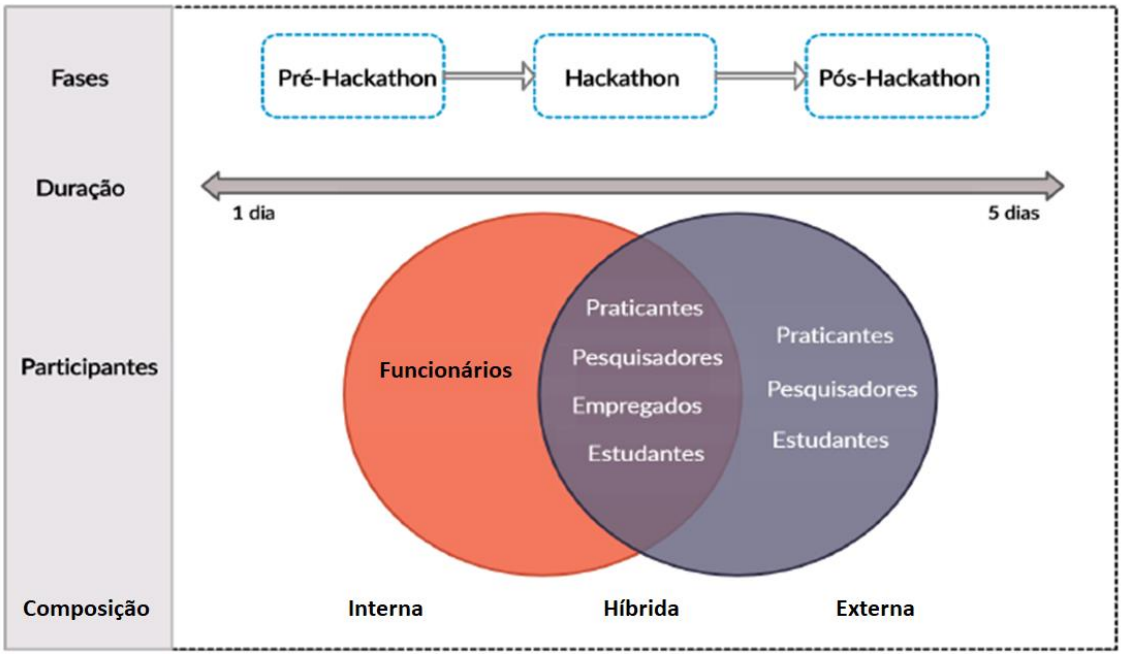

Figura 1.4. Estrutura geral de uma *hackathon* corporativa.
Fonte: (VALENÇA *et al.*, 2020a).

Outra oportunidade que as organizações devem perceber é que *hackathons* corporativas são ferramentas para (3) **difundir informação e novas tecnologias**, seja na

[8] https://miro.com/pt/

[9] https://edu.google.com/intl/ALL_br/products/jamboard/

[10] https://discord.com/

[11] https://meet.google.com/

indústria ou contexto de governo. Por exemplo, alguns eventos têm como o foco central "dados", explorando um conjunto de bases de dados da organização para criar soluções ("*datathons*"), como novas ferramentas computacionais para minerar de dados. Em outros casos, caso a organização pertença ou possua um ecossistema de software, no qual a tecnologia é uma plataforma de software (API ou SDK), cabe promovê-la. Ou seja, os resultados da organização podem ser aperfeiçoados a partir da extensão dessas tecnologias pelos participantes (público interno ou externo) ou, no mínimo, as informações acerca de sua atuação aumentam o conhecimento sobre seus processos e legitimam a sua relevância (RAATIKAINEN *et al*., 2013).

Por fim, uma recomendação adicional é entender que uma *hackathon* pode (4) **permitir a aprendizagem colaborativa** de uma tecnologia, prática ou até mesmo de um processo de negócio da organização. Esse evento pode ser usado ainda para melhorar a familiaridade dos participantes com uma ferramenta (e.g., de gestão de projetos ou de requisitos), conceito (e.g., proteção de dados ou inovação aberta) ou mesmo planejamento (e.g., analisar quais são os diferentes objetivos estratégicos). Nessa ocasião, grupos com participantes de diferentes áreas podem trocar experiências sobre esses aspectos para ampliar aspectos da gestão do conhecimento. Isso também é um trampolim para criatividade e inovação organizacional.

1.4. Cenários

Nesta seção, são discutidos dois exemplos reais de *hackathons* corporativas. A primeira, conduzida pela Airbus, foi intitulada *Airbus Systems Engineering Hackathon*. Por sua vez, a segunda foi realizada pelo *National Center for Biotechnology* (NCBI), envolvendo, como participantes, atores externos. Cada uma delas é descrita à luz do processo apresentado.

1.4.1. Airbus

O planejamento e execução desta *hackathon* foram detalhados no estudo intitulado "*A Systems Engineering Hackathon – A Methodology Involving Multiple Stakeholders to Progress Conceptual Design of a Complex Engineered Product*" (SARAVI *et al*., 2018), que é utilizado como base para ilustrar a aplicação do conjunto de fases e atividades descritas na Seção 1.3. Este evento teve como participantes os funcionários da empresa, que, durante uma semana, na sede da Airbus em Bristol (Reino Unido), tiveram como foco pensar em soluções ligadas ao projeto de aeronaves. Durante a fase ***pré-hackathon***, a equipe da Airbus encarregada da realização do evento conduziu a atividade A1 (*Definição do Objetivo e Estrutura da Hackathon*), que teve o seguinte objetivo principal como resultado: simular o projeto conceitual de uma futura e hipotética aeronave de engenharia complexa, intitulada "*Agile Wing Integration* – AWI". Entre os objetivos específicos, a equipe previu:

- Desenvolver um *framework* para futuros conceitos de operações de aeronaves;
- Reunir todas as partes interessadas (internas ou externas) envolvidas na criação de um produto, como parceiros que trabalham em diferentes aspectos do projeto, visando estabelecer o mesmo nível de entendimento sobre o escopo do projeto;

- Definir níveis adequados de granularidade de análise para cada atividade de modelagem;
- Construir um entendimento e grau de familiarização comum com os conceitos e processos de projeto de aeronaves conceituais;
- Desenvolver outros tópicos de design de asas de interesse comercial.

Ainda durante esta atividade, antes do início da *hackathon* propriamente, a Airbus identificou as partes interessadas internas e externas que estariam envolvidas no evento. A partir daí, foi possível agrupar os *stakeholders* internos em três equipes: marketing, engenharia e arquitetura. Considerando a sua ampla experiência sobre como os clientes das companhias aéreas operam seus negócios, a equipe da Airbus assumiu o papel do cliente fictício, intitulado "Companhia Aérea A".

O estudo não relata como a atividade A2 (*Organização da Infraestrutura Física e Técnica da Hackathon*) foi executada pelos organizadores da *hackathon*. No mais, cabe inferir que, por ter sido um evento envolvendo participantes internos e aberto para parceiros (e não para o público externo em geral, e.g., desenvolvedores independentes, pesquisadores, especialistas etc.), a atividade A3 (*Abertura de Inscrições da Hackathon*) não foi necessária. Também não houve resultados informados sobre a atividade A4 (*Apresentação Preliminar da Hackathon aos Participantes*), cuja realização, nesta fase inicial, não é obrigatória.

Durante a ***hackathon***, a atividade A6 (*Processo de Design e Desenvolvimento*) teve início com a divisão dos participantes em três equipes: marketing, engenharia e arquitetura, partindo da atividade A5 (*Cerimônia de Abertura da Hackathon*). A equipe de marketing buscou entender a proposta de valor da Companhia Aérea A pela visão dos seus hipotéticos passageiros. Além disso, coube a ela propor soluções que pudessem agregar valor ao negócio. Dessa forma, a equipe esteve fortemente envolvida com as fases de projeto e modelagem de soluções.

Por sua vez, a equipe de engenheiros ficou encarregada da criação de soluções técnicas baseadas em um conjunto de tecnologias modulares disponíveis para diversos componentes da aeronave, como asas e fuselagens. O objetivo da equipe era definir um conjunto de soluções viáveis e integráveis que satisfizessem os requisitos da Companhia Aérea A. Assim, a sua principal contribuição foi durante a fase de projeto conceitual, mais especificamente no dimensionamento das aeronaves. Por fim, a equipe de arquitetura, composta predominantemente por engenheiros da Airbus, atuou como uma ponte entre outras equipes, tratando de forma eficiente a relação entre as diferentes partes interessadas dos projetos. A sua principal contribuição foi propor a solução final para a Companhia Aérea A. Para isso, ela se envolveu com a maioria das fases do projeto: modelagem, análise e avaliação.

Ao final da atividade relacionada a design e desenvolvimento, os participantes, a partir de suas equipes, apresentaram como resultados um conjunto de modelos. A equipe de marketing construiu um modelo de demanda da companhia aérea, um modelo econômico e um modelo de rede. Por sua vez, a equipe de arquitetura desenvolveu um gerador de missões específicas, um modelo de custo recorrente e um modelo de desempenho. Por fim, a equipe de engenharia apresentou um gerador de conceitos e um

modelo de dimensionamento de aeronaves. Para a Airbus, esse conjunto de entregas simbolizou uma análise morfológica e um levantamento com especialistas.

O estudo não trouxe um relato das atividades de orientação (A8 – *Mentoria Técnica*) durante a condução dos projetos pelas equipes e tampouco comentou sobre a realização de uma avaliação ou premiação final (A7 – *Cerimônia de Encerramento da Hackathon*). Isto é algo que geralmente está associado a eventos abertos, com maior envolvimento e dependência de atores externos (e.g., desenvolvedores independentes atuando em uma *hackathon* baseada em uma plataforma de software).

A fase final, ***pós-hackathon***, envolveu duas atividades. A partir de uma sessão de reflexão com os participantes (A9 – *Avaliação de Satisfação dos Participantes da Hackathon*), foi possível descrever tarefas ou orientações-chave em termos de *design* para realizar uma *hackathon* bem-sucedida. Em relação aos resultados apresentados pelas equipes, houve uma verificação por especialistas da empresa, além de testes dos modelos à luz de métricas de precisão. Estas tarefas simbolizaram a preocupação da Airbus com a *Evolução das Soluções Resultantes da Hackathon* (A10).

1.4.2. National Center for Biotechnology

Esta *hackathon* foi descrita no estudo intitulado "*Closing gaps between open software and public data in a hackathon setting: user-centered software prototyping*" (BUSBY *et al*., 2016). Inicialmente, durante a ***pré-hackathon***, os organizadores definiram como principal objetivo da *hackathon* a diminuição da distância entre usuários, dados genômicos e ferramentas computacionais necessárias para analisar esses dados (A1 – *Definição do Objetivo e Estrutura da Hackathon*). Os organizadores também selecionaram problemas científicos e possíveis abordagens para resolvê-los, com base em casos de uso comuns de bioinformática e genômica que envolvem dados públicos.

Para organização da estrutura do evento (A2 – *Organização da Infraestrutura Física e Técnica da Hackathon*), a NCBI selecionou locais de realização que proporcionassem a máxima diversidade regional e acomodassem o maior número possível de participantes. Com isso, foi possível aumentar a diversidade de opiniões e habilidades. Do ponto de vista técnico, os organizadores forneceram toda a logística para desenvolvimento, como AWS Service para salvar dados, grupos do Google para comunicação e GitHub para controle de versão e desenvolvimento de software. Por fim, houve *Abertura de Inscrições da Hackathon* (A3) não só para participantes, mas também para parceiros. Ou seja, organizações locais interessadas em sediar o evento.

Embora o estudo também não apresente detalhes da ***hackathon*** em si, são descritos alguns aspectos da fase de ***pós-hackathon***. Ao final do evento, os organizadores verificaram o interesse das equipes em evoluir suas soluções com novos recursos, continuando, assim os projetos (A10 – *Evolução das Soluções Resultantes da Hackathon*). Para a NCBI, a intenção de continuidade dos projetos foi entendida como uma boa métrica para validar o sucesso da *hackathon*. Um desafio ressaltado pelos organizadores é garantir que a comunidade de usuários em potencial das soluções decorrentes da *hackathon* se envolva com a melhoria dos produtos, trazendo insumos para criação de novos módulos.

1.5. Considerações Finais

Este capítulo apresentou, como sua principal contribuição, uma perspectiva holística sobre *hackathons* corporativas, abordando conceitos básicos, bem como indicando as principais fases, atividades e características de eventos desta natureza. Essas descrições trazem luz sobre esse fenômeno e estabelecem uma estrutura para organizações que desejam planejar uma *hackathon*. Além da descrição de um processo proposto para condução de uma *hackathon* corporativa, este capítulo trouxe uma lista de boas práticas a serem consideradas em sua preparação e execução. Por fim, um cenário de aplicação real do processo proposto foi discutido a título de exemplificação a partir de dois casos reais reportados por estudos no campo.

A partir dos estudos e aplicações no âmbito de *hackathons* corporativas, algumas implicações para a teoria e prática em SI foram observadas. Primeiramente, para pesquisadores no assunto, o presente trabalho apresenta uma abordagem para *hackathons* que não é amplamente explorada na literatura, que é geralmente dedicada a eventos cívicos ou educacionais. Por sua vez, para profissionais e organizações, os resultados apresentados (i.e., o processo, com as respectivas fases e atividades, e as boas práticas) servem como uma estrutura para empresas que desejam planejar *hackathons* corporativas, além de esclarecerem detalhes de como esses eventos ocorrem atualmente. Isso considera a natureza altamente colaborativa destes eventos e a sua relevância para fomentar a criação de ecossistemas (PINHEIRO *et al*. 2020; 2021).

Como lições aprendidas para apoiar a organização de *hackathons* corporativas, foram identificadas inicialmente quatro: (1) envolver o máximo de funcionários possível, garantindo que eles pertençam a diferentes áreas ou departamentos; (2) primar por inovação aberta; (3) difundir informação e novas tecnologias; e (4) permitir a aprendizagem colaborativa. Todas essas lições se relacionam direta ou indiretamente com o conceito de inovação aberta, que tem sido explorado há alguns anos como um mecanismo para favorecer o amadurecimento e o estabelecimento das organizações no contexto de uma indústria globalizada e interconectada. Isto se relaciona de certa forma também com a necessidade de tratamento de questões econômicas e sociais nas atividades e processos relacionados ao desenvolvimento de software e serviços relacionados, apontado como um dos desafios para a indústria de software (BIFFL *et al*., 2006). Ou seja, face à quantidade de métodos, técnicas e ferramentas disponíveis no mercado, torna-se fundamental elaborar e gerir estratégias que identifiquem o valor agregado de processos/produtos e que apoiem tomadas de decisão considerando não apenas questões relativas a investimentos e custos, mas também a benefícios, riscos, oportunidades de mercado (LUZ *et al*., 2020).

Considerando a complexidade do contexto da organização de *hackathons* corporativas, que não se restringe apenas ao aspecto técnico do desenvolvimento de soluções, destaca-se ainda a importância de estudos que tentem identificar desafios e oportunidades à luz de um olhar sociotécnico (CUKIERMAN *et al*., 2007). Seja na área de SI ou de Engenharia de Software, o olhar sociotécnico visa compreender o desenvolvimento de soluções sem fragmentá-lo em fatores (ou aspectos) técnicos de um lado e fatores (ou aspectos) não-técnicos de outro. Em outras palavras, sem fatorar em quaisquer outras dualidades ("fatores técnicos" X "fatores humanos, organizacionais, éticos, políticos, sociais etc.") que terminem por desfigurar o "pano sem costura" que

imbrica o técnico e o social em um mesmo e indivisível "tecido"(SANTOS, 2010). Este olhar é importante para que a as *hackathons* corporativas sejam amadurecidas e alcancem o seu potencial de agregar valor às organizações e à indústria como um todo (TEIXEIRA & CUKIERMAN, 2007). Adicionalmente, reforça-se a necessidade de reduzir grau de incerteza sobre as decisões cotidianas, uma vez que a indústria de software apresenta uma grande velocidade que nem sempre permite uma resolução de forma estruturada, considerando dados do contexto intra- e interorganizacional, bem como experiências do mercado (FERREIRA *et al*., 2006).

Este capítulo abre oportunidades para frentes de pesquisa. Inicialmente, sugere-se uma **evolução do mapeamento sistemático apresentado em (VALENÇA *et al*., 2020a)**. O objetivo é garantir maior cobertura do conhecimento disponível sobre o assunto. Para isso, propõe-se adaptar o protocolo de pesquisa previamente elaborado para incluir também material da literatura cinzenta (e.g., *sites* com orientações sobre a condução de *hackathons*, postagens em *blogs* com relatos de eventos realizados em organizações, *tweets* etc.). *Hackathons* conduzidas na indústria são amplamente observadas em relatórios informais, como descrições de eventos conduzidos por grandes organizações, entre elas Amazon e IBM, disponíveis em seus respectivos *sites*, com informações que poderiam ser úteis para aprimorar o processo apresentado e os conceitos básicos no assunto (e.g., participantes de *hackathons*, cronogramas, objetivos etc.). O desafio de definir um processo de levantamento de dados alternativos (e.g., variedade de formas de apresentação e falta de indexação adequada dessas informações em bancos de dados) é superado pelo valor das informações da prática que podem ser obtidas, com relatos adicionais sobre *hackathons* corporativas.

Outra oportunidade de pesquisa se refere ao planejamento e condução de um **estudo experimental com foco em um conjunto de *hackathons* corporativas**. Esta investigação pode ser conduzida por meio de um estudo de caso ou etnografia e permitirá uma interação direta com especialistas na organização de *hackathons* corporativas, conforme o contexto deste tipo de evento e a forma de realizar pesquisas (ANTONIO *et al*., 2020). Assim, entrevistas e observações responderão questionamentos não explorados ou tratados pela literatura, tais como "quais das boas práticas propostas neste capítulo precisariam ser ajustadas de acordo com a cultura e localização geográfica do evento?".

Além disso, sugere-se **evoluir o macroprocesso proposto**, com o detalhamento do processo via notação BPMN (*Business Process Model and Notation*), incluindo outros elementos fundamentais à organização de uma hackathon, tais como atores de cada atividade, informações de entrada e saída, artefatos gerados e soluções tecnológicas de apoio à cada atividade, etc. Em seguida, cabe explorar e aprofundar a **estruturação de um processo configurável para organização de *hackathons* corporativas**. Nesse sentido, a partir de uma ampla coleta de dados de diferentes fontes, pretende-se desenvolver um processo cuja estrutura relacione as características apresentadas (i.e., fases, duração, participantes, composição) para derivar padrões específicos de organização de *hackathons* corporativas. Por exemplo, ao combinar a duração e a composição, será possível ajustar as fases do processo da *hackathon* (e.g., ao realizar uma *hackathon* em uma universidade, com pesquisadores e estudantes, cabe estender o total de dias do evento para favorecer a curva de aprendizado dos participantes externos e permitir resultados de maior qualidade).

Por fim, uma oportunidade adicional para avançar no estado da arte e da prática de *hackathons* corporativas envolve a adaptação do processo para domínios específicos, como eventos focados em jogos digitais. Assim, será possível, de forma empírica, analisar necessidades e particularidades de determinados cenários e propor versões do processo adaptadas a cada um deles, aumentando os benefícios em potencial do evento.

Referências

Antonio, N. P., Fornazin, M., Araujo, R. M., Santos, R. P. (2020) "Metodologia de Pesquisa - Estudo de Caso Interpretativo em Sistemas de Informação". In: J. M. David, P. M. Menezes, S. Ávila e Silva. (Org.). Tópicos em Sistemas de Informação: Minicursos SBSI 2019, SBC, pp. 53-79, doi: 10.5753/sbc.480.9.03.

Antonio, N. P., Fornazin, M., Araujo, R. M., Santos, R. P. (2021) "An Interpretative Case Study on the Scalability of Social Information Systems? The Case of "Bolsa Família" Program", iSys - Revista Brasileira de Sistemas de Informação 14(4):100-130, doi: 10.5753/isys.2021.2003.

Arndt, J. M., Dibbern, J. (2006) "Co-Innovation in a Service Oriented Strategic Network", In: Proceedings of the 2006 IEEE International Conference on Services Computing (SCC'06), Chicago, IL, USA, pp. 285-288, doi: 10.1109/SCC.2006.32.

Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., Grünbacher, P. (2006) "Value-Based Software Engineering". Springer-Verlag, doi: 10.1007/3-540-29263-2.

Boehm, B. (2006) "A View of 20th and 21st Century Software Engineering", In: Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, pp. 12-29, doi: 10.1145/1134285.1134288.

Boley, H., Chang, E. (2007) "Digital Ecosystems: Principles and Semantics", In: Proceedings of the 2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference, Cairns, QLD, Australia, pp. 398-403, doi: 10.1109/DEST.2007.372005.

Boscarioli, C., Araujo, R. M., Maciel, R. S. (2017), "I GranDSI-BR: Grand Research Challenges in Information Systems in Brazil 2016-2026", Comissão Especial de Sistemas de Informação (CESI), Sociedade Brasileira de Computação (SBC), doi: 10.5753/sbc.2884.0.

Bosch, J., Bosch-Sijtsema, P. (2010) "From Integration to Composition: On the Impact of Software Product Lines, Global Development and Ecosystems", The Journal of Systems and Software 83(1):67-76, doi: 10.1016/j.jss.2009.06.051.

Busby, B., Lesko, M., Federer, L. (2016) "Closing gaps between open software and public data in a hackathon setting: user-centered software prototyping", F1000Research 5(2016):672, doi: 10.12688/f1000research.8382.2.

Chesbrough, H. W. (2003) "Open Innovation: The New Imperative for Creating and Profiting From Technology". Harvard Business School Press.

Costa, L. A., Fontão, A., Santos, R. P. (2021) "Toward Proprietary Software Ecosystem Governance Strategies Based on Health Metrics", IEEE Transactions on Engineering Management, doi: 10.1109/TEM.2021.3116531.

Cukierman, H. L., Teixeira, C., Prikladnicki, R. (2007) “Um Olhar Sociotécnico sobre a Engenharia de Software”, Revista de Informática Teórica e Aplicada 14(2):207-227, doi: 10.22456/2175-2745.5696.

Ferreira, C. A., Werner, C. M. L., Barros, M. O. (2006) “Gerência de Carteiras de Componentes: Uma Abordagem Baseada em Valor”, In: Anais do VI Workshop de Desenvolvimento Baseado em Componentes (WDBC), Recife, Brasil, pp. 22-29.

Fontão, A., Cleger-Tamayo, S., Wiese, I., Santos, R. P., Dias-Neto, A. C. (2021) “A Developer Relations (DevRel) model to govern developers in Software Ecosystems”, Journal of Software-Evolution and Process, e2389, doi: doi.org/10.1002/smr.2389.

Goldman, R., Gabriel, R. P. (2005) “Innovation Happens Elsewhere: Open Source as Business Strategy”. Morgan Kaufmann.

Graciano Neto, V V., Santos, R. P., Viana, D., Araujo, R. (2020) “Towards a Conceptual Model to Understand Software Ecosystems Emerging from Systems-of-Information Systems”, In: Santos R., Maciel C., Viterbo J. (eds) Software Ecosystems, Sustainability and Human Values in the Social Web. WAIHCWS 2017, WAIHCWS 2018. Communications in Computer and Information Science, vol 1081. Springer, Cham. https://doi.org/10.1007/978-3-030-46130-0_1.

Hanssen, G. K. (2012) “A Longitudinal Case Study of an Emerging Software Ecosystem: Implications for Practice and Theory”, The Journal of Systems and Software 85(7):1455-1466, doi: 10.1016/j.jss.2011.04.020.

Herala, A., Kokkola, J., Kasurinen, J., Vanhala, E. (2019) “Strategy for data: Open it or hack it?”, Journal of Theoretical and Applied Electronic Commerce Research 14(2): 33-46, https://dl.acm.org/doi/abs/10.5555/3289217.3289221.

Iansiti, M., Levien, R. (2004) “Strategy as Ecology”, Harvard Business Review 82(3):68-78, 126, PMID: 15029791, https://hbr.org/2004/03/strategy-as-ecology.

Kimbell, L. (2011) “Rethinking Design Thinking: Part I”, Design and Culture 3(3):285-306, doi: 10.2752/175470811X13071166525216.

Luz, P. B. V., Fernandes, J., Valença, G., Santos, R. P. (2020) “Exploring Sustainability in Real Cases of Emerging Small-to-Medium Enterprises Ecosystems”, In: Santos R., Maciel C., Viterbo J. (eds) Software Ecosystems, Sustainability and Human Values in the Social Web. WAIHCWS 2017, WAIHCWS 2018. Communications in Computer and Information Science, vol 1081. Springer, Cham, doi: 10.1007/978-3-030-46130-0_3.

Nolte, A., Pe-Than, E. P. P., Filippova, A., Bird, C., Scallen, S., Herbsleb, J. D. (2018) “You Hacked and Now What? Exploring Outcomes of a Corporate Hackathon”, In: Proceedings of the ACM Human-Computer Interaction 2, CSCW, Article 129 (November), 23 pages, doi: 10.1145/3274398.

Pe-Than, E. P P., Nolte, A., Filippova, A., Bird, C., Scallen, S., Herbsleb, J. D. (2019) “Designing Corporate Hackathons with a Purpose: The Future of Software Development”, IEEE Software 36(1):15-22, doi: 10.1109/MS.2018.290110547.

Pinheiro, M. C., Chueri, L. O. V., Santos, R. P. (2020) “Identifying Topics and Difficulties on Collaboration in Social Innovation Environments”, In: Proceedings of

the XVI Brazilian Symposium on Information Systems (SBSI'20), São Bernardo do Campo, Brazil, pp. 1-8 (Article 5), doi: 10.1145/3411564.3411581.

Pinheiro, M. C., Chueri, L. O. V., Santos, R. P. (2021) "Investigando Colaboração em Ecossistemas", In: Anais do VI Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES), Florianópolis, Brasil, pp. 11-20, doi: 10.5753/washes.2021.15885.

Porras, J., Khakurel, J., Ikonen, J., Happonen, A., Knutas, A., Herala, A., "Hackathons in Software Engineering Education - Lessons Learned from a Decade of Events", In: Proceedings of the 2018 IEEE/ACM International Workshop on Software Engineering Education for Millennials (SEEM), Gothenburg, Sweden, pp. 40-47, https://ieeexplore.ieee.org/document/8442127.

Raatikainen, M., Komssi, M., Bianco, V. d., Kindstöm, K., Järvinen, J. (2013) "Industrial Experiences of Organizing a Hackathon to Assess a Device-centric Cloud Ecosystem", In: Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference, Kyoto, Japan, pp. 790-799, doi: 10.1109/COMPSAC.2013.130.

Rosell, B., Kumar, S., Shepherd, J. (2014) "Unleashing innovation through internal hackathons", In: Proceedings of the 2014 IEEE Innovations in Technology Conference, Warwick, RI, USA, pp. 1-8, doi: 1.1109/InnoTek.2014.6877369.

Santos, R. P. (2010) "Brechó-VCM: Uma Abordagem Baseada em Valor para Mercados de Componentes". Dissertação, Mestrado em Engenharia de Sistemas e Computação, COPPE/UFRJ, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, doi: 10.13140/RG.2.2.32705.07525.

Santos, R. P. (2013) "Engenharia e Gerenciamento de Ecossistemas de Software". Exame de Qualificação, Doutorado em Engenharia de Sistemas e Computação, COPPE/UFRJ, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, http://reuse.cos.ufrj.br/media/publicacoes/qualificacao/EQ_RodrigoSantos.pdf.

Santos, R. P. (2016) "Managing and Monitoring Software Ecosystem to Support Demand and Solution Analysis". Tese, Doutorado em Engenharia de Sistemas e Computação, COPPE/UFRJ, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, doi: 10.13140/RG.2.2.13391.18082.

Santos, R. P., Fontão, A. L., Fernandes, J. C. (2020) "Ecossistemas de Software". In: Maciel, C.; Viterbo, J. (Org.). Computação e Sociedade: A Tecnologia, EdUFMT, v. 3, pp. 198-223.

Saravi, S., Joannou, D., Kalawsky, R. S., King, M. R. N., Marr, I., Hall, M., Wright, P. C. J., Ravindranath, R., Hill, A. (2018) "A Systems Engineering Hackathon – A Methodology Involving Multiple Stakeholders to Progress Conceptual Design of a Complex Engineered Product", IEEE Access 6(2018):38399-38410, doi: 10.1109/ACCESS.2018.2851384

Steglich, C., Santos, R. P., Marczak, S., Perin, M., Mosmann, L. H., Guerra, L. P., Souza, C. R. B., Figueira Filho, F. (2021) "An Investigation of the Strengths, Weaknesses, Opportunities and Threats in the Business Dimension for Develpers in

Mobile Software Ecosystems", iSys - Revista Brasileira de Sistemas de Informação 14(4):74-99, doi: 10.5753/isys.2021.2015.

Teixeira, C. A. N., Cukierman, H. L. (2007) "Por que Falham os Projetos de Implantação de Processos de Software?", In: Anais do III Workshop Um Olhar Sociotécnico sobre a Engenharia de Software (WOSES), em conjunto com VI Simpósio Brasileiro de Qualidade de Software, Porto de Galinhas, Brasil, pp. 1-12.

Valença, G., Lacerda N., Rebelo M. E., Alves C., de Souza C. R. B. (2019) "On the Benefits of Corporate Hackathons for Software Ecosystems – A Systematic Mapping Study". In: Franch X., Männistö T., Martínez-Fernández S. (eds) Product-Focused Software Process Improvement. PROFES 2019. Lecture Notes in Computer Science, Springer, v. 11915, pp. 367-382, doi: 10.1007/978-3-030-35333-9_27.

Valença, G., Lacerda, N., de Souza, C. R. B., Gama, K. (2020) "A Systematic Mapping Study on the Organisation of Corporate Hackathons", In: Proceedings of the 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Portoroz, Slovenia, pp. 421-428, doi: 10.1109/SEAA51224.2020.00074.

Valença, G., Kneuper, R., Rebelo, M. (2020) "Privacy in Software Ecosystems-An Initial Analysis of Data Protection Roles and Challenges", In: Proceedings of the 2020 46th Euromicro Conference on Software Engineering and Advanced Applications, Portoroz, Slovenia, pp. 120-123, doi: 10.1109/SEAA51224.2020.00028.

Capítulo 2

Deep Learning para Processamento de Linguagem Natural

Eduardo Soares de Paiva e Fernando Sola Pereira

Abstract

This chapter presents an overview of the main natural language processing techniques currently used. Studies demonstrate evolutions in textual representation and texts processing. For text representation, we demonstrated techniques to transform texts in numerical formats and to capturing semantic and syntactic information of words. Text processing researches also point to the use of deep learning algorithms. Thus, this chapter analyzes the forms of text representation, RNN and CNN networks for natural language processing, transformers architecture and the BERT model.

Resumo

Esse capítulo apresenta uma visão sobre as principais técnicas de processamento de linguagem natural utilizadas atualmente. Estudos demonstram evoluções tanto nas formas de representação textual, quanto no modo de processamento dos textos. Para a representação de texto, apresentamos as técnicas utilizadas para transformar textos em formatos numéricos e para capturar informações semânticas e sintáticas de palavras. Pesquisas em processamento de texto também apontam para o uso de algoritmos de aprendizado profundo. Sendo assim, esse trabalho analisa as formas de representação textual, as redes RNN e CNN para processamento de linguagem natural, a arquitetura transformers e o modelo BERT.

2.1. Introdução

Esse capítulo apresenta uma visão sobre as principais técnicas de processamento de linguagem natural com a utilização de *deep learning*.

Atualmente, a maioria dos dados disponíveis para análise encontra-se em formato textual. Logo, essa é uma área de pesquisa que tem recebido bastante atenção nos últimos anos. As atividades de processamento de linguagem natural podem ser utilizadas

para classificação textual [Paiva et al., 2021], análise de sentimentos [Yadav and Vishwakarma, 2020], tradução automática de textos [Hassan et al., 2018], sumarização de textos [Kosmajac and Kešelj, 2019], dentre outras.

Estudos demonstram evoluções nas formas de representação textual, e no modo de processamento dos textos. Quanto a representação textual, foram desenvolvidas técnicas capazes não só de representar os textos, como também de capturar informações semânticas e sintáticas das palavras [Mikolov et al., 2013b], [Mikolov et al., 2013a], [Pennington et al., 2014], [Devlin et al., 2019].

Já as pesquisas na área de processamento dos textos têm apontado para a utilização de algoritmos de deep learning (redes neurais profundas). Nesse contexto, as redes *Convolutional Neural Network-CNN* [Lecun et al., 1998], *Recurrent Neural Network-RNN* [Elman, 1990] e as baseadas na arquitetura Transformers [Vaswani et al., 2017] têm demonstrado bons resultados.

Sendo assim, esse trabalho faz um levantamento de tais técnicas a fim de fornecer um panorama sobre os principais direcionamentos utilizados no processamento de textos.

O restante desse Capítulo está dividido da seguinte forma: a Seção 2.2 apresenta um estudo sobre representação de palavras. As Seções 2.3, 2.4 e 2.5 descrevem as redes neurais convolucionais, as redes neurais recorrentes e a arquitetura *transformer*, respectivamente. Já a Seção 2.6 demonstra o funcionamento e utilização da arquitetura *Bidirectional Encoder Representations from Transformers* (BERT) e a Seção 2.7 apresenta outros modelos que tentam tratar limitações do modelo BERT. Finalmente, a Seção 2.8 faz uma breve conclusão do estudo.

2.2. Representação de Palavras

Os modelos de redes neurais não trabalham com o texto bruto, eles esperam como entrada vetores numéricos [Chollet, 2017]. Logo, antes de se iniciar o processamento de textos com redes neurais, realiza-se um processo de vetorização textual.

Atualmente, a forma mais usual de se representar palavras como vetores numéricos é pela utilização de *word embeddings*. *Word embedding* é uma forma de representação textual que utiliza vetores densos[1], de baixa dimensionalidade, cujos valores são aprendidos a partir do próprio texto. A *word embedding* utiliza a vizinhança de cada uma das palavras do texto para formular a representação das palavras.

Isso permite a criação de um vetor denso que representa a projeção de cada palavra. Dessa forma, a *word embedding* representa as coordenadas da palavra no espaço vetorial que foi aprendido a partir do texto. Sendo assim, os relacionamentos geométricos entre os vetores de palavras devem refletir os relacionamentos semânticos entre essas palavras [Goyal et al., 2018]. Utilizando-se essa abordagem, pode-se comparar a relação entre duas palavras quaisquer a partir da comparação dos vetores que as representam.

Esse tipo de representação surgiu da necessidade de se expressar as características de similaridade entre as palavras, de forma que isso pudesse ser aproveitado no contexto

[1]Um vetor denso é vetor cujas posições são preenchidas com valores diferentes de zero. Em contrapartida, um esparso é aquele cuja maioria das posições são preenchidas com o valor zero.

das aplicações.

Sendo assim, passou-se a explorar o conceito de modelagem estatística da linguagem. Esse modelo permite a previsão da palavra seguinte, levando-se em consideração as anteriores [Bengio et al., 2000]. Essa ideia é uma extensão dos modelos utilizados no tratamento de séries temporais, pois, da mesma forma que em sistemas lineares, o estado seguinte pode ser determinado pela combinação dos estados anteriores. Logo, pode-se prever a palavra seguinte de um determinado texto, com base nas palavras que ocorreram anteriormente nesse mesmo texto.

Essa constatação já havia sido feita por linguistas, Harris (1954) já afirma que as palavras vizinhas estavam relacionadas semanticamente, logo, palavras semelhantes ocorreriam em contextos semelhantes. Nesse mesmo sentido, Firth (1957) afirma que é possível conhecer uma palavra através das palavras que a acompanham (*"You shall know a word by the company it keeps!"*). Dessa forma, conclui-se que as palavras não ocorrem em contextos independentes, uma palavra sempre está relacionada com as palavras que vêm antes e depois.

Sendo assim, pode-se destacar duas características importantes do modelo de linguagem:

- A probabilidade de se encontrar uma palavra em um determinado texto é função da ocorrência de todas as palavras anteriores;
- Uma palavra sempre está relacionada com seus vizinhos.

Logo, a partir dessas premissas, formulou-se a seguinte expressão, indicada na Equação 1, para representar a probabilidade de ocorrência de uma determinada palavra em uma sequência de dados textuais. Ou seja, para se obter a probabilidade de ocorrência de uma palavra localizada na posição t, deve-se considerar a probabilidade de ocorrência de todas as palavras que apareceram nas $t-1$ posições anteriores (razão pela qual o valor de K, apresentado na Equação 1, varia entre 1 e $t-1$).

$$p(w^{(t)}) = \prod_{k=1}^{k=t-1} p(w^{(k)}|\{w^1,...,w^{k-1}\}) \qquad [Bengioet\ al.,\ 2000] \qquad (1)$$

Logo, a probabilidade de ocorrência de uma determinada palavra em um texto será igual ao produto das probabilidades de ocorrência das palavras que ocorreram antes dela nesse mesmo texto.

No entanto, essa é apenas uma formulação teórica, pois na maioria das situações não é possível utilizar todas as palavras anteriores (desde o início do texto) para se prever a próxima. Então, para tornar essa premissa viável de ser aplicada em situações práticas, faz-se uma aproximação, e em vez de se considerar todas as palavras para o cálculo da probabilidade da palavra seguinte, realiza-se esse cálculo com base em uma janela de tamanho fixo. Ou seja, calcula-se a probabilidade de uma palavra com base nas n palavras anteriores (onde n é o tamanho da janela a ser considerada). Dessa forma, a Equação 1 pode ser simplificada para a Equação 2:

$$p(w^{(t)}) \approx \prod_{k=t-n+1}^{k=t-1} p(w^{(k)}|\{w^{k-1},...,w^{k-n}\}) \qquad [Bengioet\ al.,\ 2000] \qquad (2)$$

Isso permite a modelagem de qualquer palavra do texto em função de um número fixo de parâmetros, o que torna tal representação muito mais concisa. A partir dessa consideração, torna-se possível desenvolver uma série de modelos de *word embeddings* que são utilizados para modelagem textual. Existem vários modelos de *word embeddings* que se utilizam dessa premissa. Esse estudo apresenta dois exemplos desses tipos de modelos: word2Vec [Mikolov et al., 2013b] e [Mikolov et al., 2013a] e GloVe [Pennington et al., 2014].

2.2.1. Word2Vec

Apesar de algumas tentativas anteriores, as primeiras iniciativas que apresentaram resultados satisfatórios, para a representação de palavras como vetores densos, foram as propostas em Mikolov et al. [2013b] e Mikolov et al. [2013a]. Esses trabalhos apresentam a representação distribuída de palavras obtida a partir da utilização de uma rede neural de duas camadas. Tal modelo ficou conhecido como o Word2Vec, dividindo-se em duas variantes: skip-gram e *Continuous Bag of Words* (CBOW).

- **Skip-gram**

O modelo skip-gram [Mikolov et al., 2013a] é a variante do word2vev que prevê o contexto baseado na palavra corrente, ou seja, dada uma palavra de entrada, ele tenta prever as palavras que aparecem antes e depois dessa palavra de entrada. Para isso, utiliza-se uma rede neural de duas camadas que é treinada com um determinado corpus, ou seja, o próprio texto já oferece a entrada (uma palavra central) e os valores de referência que devem ser utilizados no processo de treinamento (palavras vizinhas), constituindo-se assim em um processo auto supervisionado.

Logo, a entrada dessa rede neural é um vetor cujo tamanho é a palavra central, e a saída são as palavras vizinhas que se pretende prever. Nessa situação, a *word embedding* é representada pelos pesos da camada escondida da rede neural.

A Figura 2.1 ilustra a estrutura da rede neural utilizada para encontrar a representação das palavras empregando a variante skip-gram do modelo word2vec, sendo que, nesse caso, V é o tamanho do vocabulário[2], N é a quantidade de neurônios da camada escondida, x e y são vetores *one-hot encoding*[3] que representam as entradas e saídas do modelo, respectivamente, W_{VN} é a matriz de pesos entre a camada de entrada e a camada escondida, onde a i-esima linha da matriz representa o peso correspondente da i-esima palavra do vocabulário, a matriz W'_{VN} representa os pesos entre a camada escondida e a camada de saída e C está relacionado ao tamanho do contexto (quantidade de vizinhos da palavra alvo). Os vetores da word embedings estão contidos na matriz de pesos W_{VN}.

[2] Vocabulário: conjunto de palavras (ou tokens) que devem ser consideradas no processamento do texto.

[3] Vetor *one-hot encoding* é um vetor em que todas as suas posições são preenchidas com o valor zero, exceto a posição da palavra que ele está representando, que é preenchida com o valor 1.

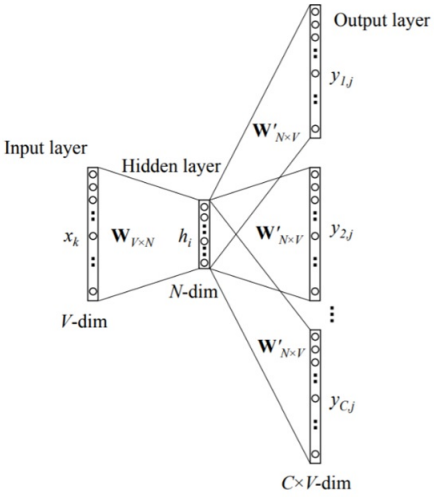

Figura 2.1. Estrutura de Funcionamento do word2vec - skip-gram. Fonte: [Rong, 2014].

Sendo assim, considerando a frase *"esse é um exemplo de sentença"*, dada a palavra central *"exemplo"*, o modelo tenta prever as palavras do contexto: *"esse"*, *"é"*, *"um"*, *"de"* e *"sentença"*.

O algoritmo funciona da seguinte forma: primeiro gera-se o vetor *one-hot encoding* da palavra central, depois obtém-se o vetor de pesos correspondente à palavra em questão (*word embeding* dessa palavra central). O passo seguinte é a geração do *score z*, sendo que esse *score* é obtido pela multiplicação do vetor de entrada pelo vetor de pesos (como a entrada é um vetor esparso, *one-hot encoding*, quando é feita a multiplicação da entrada pela matriz de pesos, apenas a linha correspondente à palavra de entrada em questão é considerada). Posteriormente é feito o cálculo da probabilidade das palavras de saída, para isso, utiliza-se a função softmax[4] ($y = softmax(z)$). Depois, compara-se o vetor de probabilidades geradas com o vetor de probabilidades reais. Isso possibilita a atualização dos pesos pelo método do gradiente e repete-se esse processo de forma iterativa até o fim do treinamento.

A probabilidade, obtida pela aplicação da função softmax, é dada pela Equação 3. Nessa Equação, υ_O indica a representação da palavra de saída ($w^{(t)}$) e μ_I é a representação da palavra de entrada ($w^{(t+k)}$).

$$p(w^{(t+k)}|w^{(t)}) = p(\upsilon_O|\mu_I) = \frac{exp(\upsilon_O^T \mu_I)}{\sum_{j=1}^{M} exp(\upsilon_j^T \mu_I)} \qquad [Mikolovet\ al.,\ 2013b] \quad (3)$$

Logo, obtêm-se a probabilidade de uma determinada palavra $(t+k)$, a partir de uma palavra t, pela utilização da função softmax aplicada ao produto de dois vetores u e v.

[4] A função softmax recebe como entrada um vetor de valores reais e fornece como saída um vetor com valores entre 0 e 1 e a soma desses valores é igual a 1.

- ***Continuous Bag of Words*** **(CBOW)**

O modelo CBOW [Mikolov et al., 2013a] funciona de forma análoga ao skip-gram. No entanto, nessa versão do word2vec a representação da palavra é obtida pelo ajuste dos parâmetros do modelo que tenta fazer a previsão de uma determinada palavra a partir do seu contexto (palavras que aparecem na sua vizinhança), conforme apresentado na Figura 2.2.

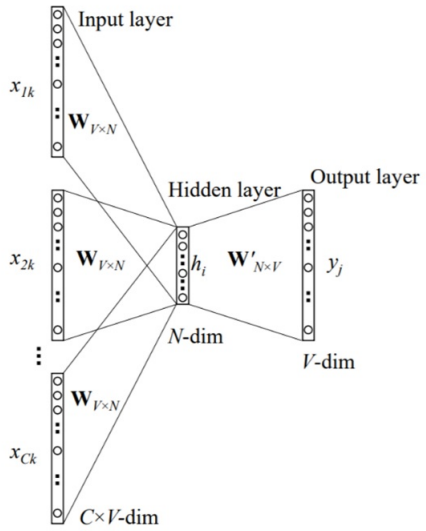

Figura 2.2. Estrutura de Funcionamento do word2Vec – CBOW. Fonte: [Rong, 2014].

Sendo assim, considerando a frase *"esse é um exemplo de sentença"*, dadas as palavras do contexto: *"esse"*, *"é"*, *"um"* e *"de"* e *"sentença"*, o modelo tenta prever a palavra central *"exemplo"*.

Nesse caso, as entradas são vetores *one-hot encoding*, que representam cada uma das palavras de contexto, e a saída é o vetor *one-hot encoding* que faz a previsão da palavra central. Cabe ressaltar que o algoritmo do CBOW funciona de forma análoga ao skip-gram, sendo que, a única diferença é que no caso do CBOW as entradas e saídas da rede neural são invertidas. Ou seja, as palavras de contexto funcionam como entrada da rede neural, e a palavra central é a saída da rede. Logo, essa palavra central funcionará como parâmetro de comparação para se realizar o ajuste dos pesos.

2.2.2. GloVe

O modelo Global Vectors for Word Representation - GloVe [Pennington et al., 2014] trabalha com as probabilidades de coocorrência de palavras em um texto para incorporá-las em vetores significativos. Ou seja, ele considera a frequência em que uma palavra j aparece no contexto de uma palavra i em todo o texto. Dessa forma, considera-se X a matriz de coocorrência e X_{ij} o número de vezes que a palavra j aparece no contexto da palavra i.

A probabilidade de coocorrência de uma palavra j ocorrer com uma palavra i é a razão entre o número de vezes que a palavra j ocorre no contexto da palavra i sobre o

número de vezes que qualquer palavra aparece no contexto da palavra *i*, conforme apresentada na Equação 4.

$$P_{ij} = P(j|i) = \frac{X_{ij}}{\sum_{i \in context} X_{ik}} \qquad [Penningtonet\ al.,\ 2014] \tag{4}$$

Dessa forma, o GloVe verifica a razão entre as probabilidades de coocorrências para extrair o significado interno das palavras. Pennington et al. (2014) utilizam a Tabela 2.1 para exemplificar essa ideia.

Probabilidade e Razão	k = solid	k = gas	k = water	k = fashion
P(k/ice)	$1,9 \times 10^{-4}$	$6,6 \times 10^{-5}$	$3,0 \times 10^{-3}$	$1,7 \times 10^{-5}$
P(k/steam)	$2,2 \times 10^{-5}$	$7,8 \times 10^{-4}$	$2,2 \times 10^{-3}$	$1,8 \times 10^{-5}$
P(k/ice)/P(k/steam)	8,9	$8,5 \times 10^{-2}$	1,36	0,96

Tabela 2.1. Probabilidades de Coocorrências [Pennington et al., 2014]

Pennington et al. (2014) explicam que as duas primeiras linhas da tabela mostram as probabilidades das palavras *"solid"*, *"gas"*, *"water"* e *"fashion"* ocorrerem no contexto das palavras *"ice"* e *"steam"*. Já a última linha mostra a razão de probabilidades.

Para palavras relacionadas a *"ice"*, mas não a *"steam"*, como *"solid"*, a razão é alta. Por outro lado, para palavras relacionadas a *"steam"*, mas não a *"ice"*, como *"gas"*, a razão é baixa e, para palavras relacionadas a ambos ou a nenhum deles, como *"water"* e *"fashion"*, respectivamente, a razão é próxima de 1.

Dado que as razões de probabilidades de coocorrência capturam informações relevantes sobre o relacionamento das palavras, o modelo GloVe visa obter uma função F que prevê essas proporções a partir de dois vetores de palavras w_i e w_j e de um vetor de palavra de contexto w_k, conforme apresentado na Equação 5.

$$F(w_i, w_j, \tilde{w_k}) = \frac{P_{ik}}{P_{jk}} \qquad [Penningtonet\ al.,\ 2014] \tag{5}$$

Sendo assim, o modelo aprende as representações dos vetores de palavras w_i, w_j e w_k para alimentar a função F e prever corretamente as proporções de probabilidades.

O GloVe prediz as palavras circundantes maximizando a probabilidade de uma palavra de contexto ocorrer, a partir de uma palavra central, utilizando para isso uma regressão logística.

2.3. *Convolution Neural Network*

As redes neurais convolucionais (CNNs) são largamente empregadas em atividades de visão computacional, obtendo excelentes resultados em tais tarefas [Krizhevsky et al., 2017], [Han et al., 2018]. Esse sucesso se dá principalmente pelo fato do seu processamento ser realizado de forma convolucional[5], o que é muito eficaz para casos em que os

[5]A convolução é uma operação matemática entre duas funções g e h, produzindo uma terceira função que normalmente é vista como uma versão modificada de uma das funções originais[Díaz-Guerra et al.,

dados tenham alguma relação topológica com os seus vizinhos, como ocorre em imagens. Tal propriedade permite a extração de características que são capazes de definir os dados que são utilizados como entrada.

Essa mesma propriedade, que torna as CNNs tão eficazes para tarefas de visão computacional, também é útil para a extração de características importantes em dados sequenciais, pois, via de regra, os dados pertencentes a uma sequência textual também compartilham características com seus elementos vizinhos. Um texto pode ser visto como uma sequência de palavras. Logo, assim como em uma imagem um pixel compartilha algumas características em comum com os seus pixels vizinhos, as palavras de um texto também guardam algum tipo de relação com as demais palavras que aparecem nas suas proximidades. Sendo assim, pode-se utilizar esse tipo de rede para treinar modelos capazes de desempenhar tarefas de processamento de linguagem natural, conforme pode ser observado em [Kim, 2014], [Zhang et al., 2015] e [Johnson and Zhang, 2015].

No entanto, enquanto nas tarefas de visão computacional os dados são tratados de forma bidimensional (largura e altura da imagem), nas tarefas de Processamento de Linguagem Natural com CNNs, os dados aparecem de forma unidimensional, onde a única dimensão tratada é a dimensão temporal. Logo, cada palavra do texto é considerada como o valor correspondente a um determinado instante de tempo. Sendo assim, para o caso de processamento de textos, ao invés de se utilizar convoluções 2D (como é o caso do processamento de imagens), utiliza-se convolução 1D.

A Figura 2.3 ilustra o processo de convolução 1D para uma determinada sequência de tokens. A convolução começa capturando os primeiros tokens da sequência da entrada, considerando-se o tamanho do kernel do filtro. A partir dessa captura, faz-se a multiplicação desses valores pelos filtros e obtém-se a saída correspondente a essa parte da entrada, sendo que esse processo se repete até que se chegue ao final da sequência de entrada. A CNN unidimensional é invariante para translações (assim como a bidimensional), o que significa que certas sequências podem ser reconhecidas mesmo que apareçam em posições diferentes. Isso pode ser útil para a identificação de determinados padrões no texto.

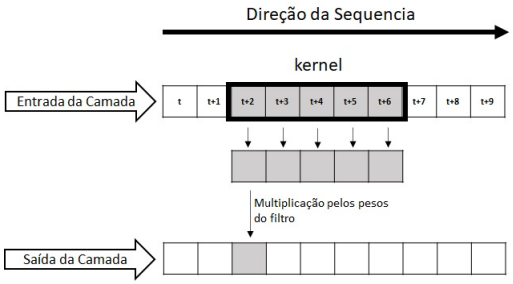

Figura 2.3. Processo de Convolução 1D. Fonte: Elaborada pelos autores.

Essa capacidade das camadas de convolução 1D reconhecerem padrões locais em uma sequência se dá pelo fato de que a mesma transformação de entrada é executada em cada janela. Logo, o padrão aprendido em uma determinada posição de uma sentença

2012].

pode ser reconhecido posteriormente em uma posição diferente. Por essa razão, diz-se que a convolução é invariante a translações. Dessa forma, caso se tenha um filtro sensível[6] a uma determinada sequência de palavras, ele será ativado sempre que tal sequência aparecer no texto, independentemente de sua posição dentro da frase.

Associada à camada de convolução, utiliza-se uma camada de *pooling*, cujo objetivo é diminuir a dimensionalidade dos dados de entrada e tornar o modelo pouco sensível a operações de rotação e de translação, fazendo uma espécie de subamostragem. A operação de *pooling* tem algumas variantes no critério de subamostragem, como por exemplo, *pooling* pela média ou *pooling* pelo valor máximo.

Sendo assim, as camadas de convolução e de *pooling* trabalham de forma combinada, com o objetivo de extrair um conjunto de características capazes de identificar algumas propriedades do conjunto de dados de entrada.

2.4. *Long Short-Term Memory Networks*

Uma das principais características das redes neurais convencionais (*feedforward*) é que elas não têm memória. Cada entrada apresentada a elas é processada independentemente, sem que nenhum estado seja mantido entre as entradas. Dessa forma, para se processar uma sequência de dados, é necessário mostrar a sequência inteira à rede de uma só vez, transformando-a em um único ponto de dado.

No entanto, muitas vezes é importante se ter alguma informação a respeito do que foi processado nos instantes anteriores, para utilizá-la no processamento atual. Essa característica é especialmente importante para o processamento de textos, visto que, muitas das vezes o significado de uma determinada palavra depende das palavras que foram processadas anteriormente.

As redes neurais recorrentes (*recurrent neural network* - RNN) [Elman, 1990] surgiram com o objetivo de preencher essa lacuna. Esse tipo de rede neural processa sequências de dados e mantêm as informações referentes a esse processamento. Nas RNNs há uma espécie de realimentação a partir da sua saída, o que permite com que as variáveis de estado possam ser mantidas para auxiliar nas próximas previsões.

No entanto, as redes neurais recorrentes tradicionais apresentam o problema da dissipação do gradiente (*vanish gradient*) e da explosão do gradiente [Pascanu et al., 2013]. Esse tipo de problema dificulta o processo de treinamento da rede, pois, durante a fase de retropropagação do gradiente, o sinal do gradiente acaba sendo multiplicado várias vezes pela matriz de peso associada às conexões entre os neurônios da camada recorrente. Isso faz com que o valor dessas multiplicações tenda para zero ou para valores muito altos, o que, em ambos os casos, é prejudicial para o processo de treinamento da rede.

Pascanu et al. (2013) demonstram esse problema. Basicamente, se a matriz for formada por valores pequenos (se o autovalor principal da matriz de peso for menor que 1), isso fará com que o valor do gradiente fique tão pequeno a ponto de paralisar o processo de treinamento. Esse fenômeno é conhecido como dissipação do gradiente

[6]Nesse contexto, entende-se como um filtro sensível, aquele que é capaz de capturar algum tipo específico de padrão nos dados que estão sendo processados.

(ou *vanishing gradient*). Por outro lado, se os pesos dessa matriz forem altos (autovalor principal da matriz de peso maior que 1), o sinal de gradiente pode ficar tão grande a ponto de fazer com que o processo de treinamento não convirja. Esse fenômeno é conhecido como explosão do gradiente (ou *exploding gradient*).

Essas limitações motivaram o surgimento das redes LSTM [Hochreiter and Schmidhuber, 1997]. Os neurônios de uma camada LSTM possuem uma célula de memória que é composta por quatro elementos principais: um portão de entrada, um portão com uma conexão auto-recorrente, um portão de esquecimento e um portão de saída. A conexão auto-recorrente tem um peso de 1 e permite que o estado da célula de memória permaneça constante de um passo para outro. Os portões servem para regular as interações entre a própria célula de memória. A porta de entrada pode permitir que o sinal recebido altere o estado da célula de memória ou bloqueie-o. Já a porta de saída pode permitir que o estado da célula de memória tenha efeito sobre outros neurônios ou não. O portão de esquecimento trata a conexão recorrente da célula de memória, permitindo que a célula se lembre ou esqueça do seu estado anterior, conforme necessário. A Figura 2.4 ilustra uma célula LSTM.

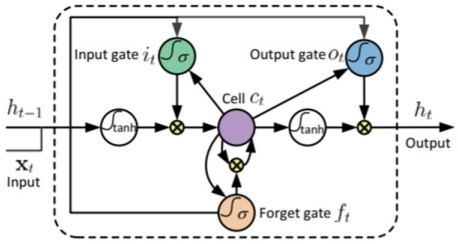

Figura 2.4. Neurônio LSTM. Fonte: [Zhu et al., 2016].

Esse tipo de rede não sofre com os problemas da degradação e explosão do gradiente pelo fato de se gerar um fluxo contínuo para o cálculo do gradiente, pois, o cálculo do gradiente em relação ao espaço interno de memória não é retropropagado para esses parâmetros. Dessa forma, o erro retropropagado passa direto pela camada, não desaparecendo, nem explodindo, para sequências mais longas. Essa característica faz com que essa topologia melhore os resultados quando se precisa de memórias mais longas[7].

Por tais características, as redes LSTMs vêm sendo largamente utilizadas nas tarefas de processamento de linguagem natural, pois a topologia da rede LSTM permite que sejam guardadas algumas informações referentes as palavras anteriores, sem sofrer os problemas de treinamento das RNNs tradicionais, o que pode ser útil para o processamento da palavra atual.

2.5. Arquitetura *Transformer*

Outra arquitetura de rede que tem ganhado relevância no contexto de processamento de linguagem natural é a arquitetura transformer, que é composta por duas partes: um codi-

[7]Entende-se como necessidade de memórias mais longas as situações em que o texto processado é muito grande e as células de memória precisam armazenar muitas informações relativas as palavras anteriores.

ficador e um decodificador.

Porém, o conceito de codificadores e decodificadores é anterior à arquitetura *transformer*, um exemplo disso é a arquitetura proposta em [Cho et al., 2014]. Cho et al. (2014) propõem duas redes neurais recorrentes (RNN) que atuam como um codificador e um decodificador.

O codificador mapeia uma sequência fonte de comprimento variável para um vetor de comprimento fixo e o decodificador mapeia essa representação vetorial de volta para uma sequência alvo de comprimento variável.

Na arquitetura proposta por [Cho et al., 2014], as duas redes (codificador e decodificador) são treinadas em conjunto para maximizar a probabilidade condicional da sequência de destino, dada uma sequência de origem.

Nessa arquitetura, o codificador é responsável por receber sequências textuais e o decodificador utiliza a saída desse codificador para produzir respostas para os textos de entrada. Esse princípio é útil para problemas de tradução automática de textos, pois, nesse tipo de problema, deseja-se mapear uma sequência de palavras (que forma uma sentença) em uma outra sequência de palavras que forma uma sentença com a mesma carga semântica da sequência de entrada, mas escrita em outro idioma.

Utilizando esse princípio de codificadores e decodificadores, Vaswani et al. (2017) propõem uma nova arquitetura, denominada *transformers*, que é centrada em mecanismos de atenção. A Figura 2.5 apresenta a arquitetura *transformers*, sendo que, a parte esquerda representa o codificador e a parte direita representa o decodificador.

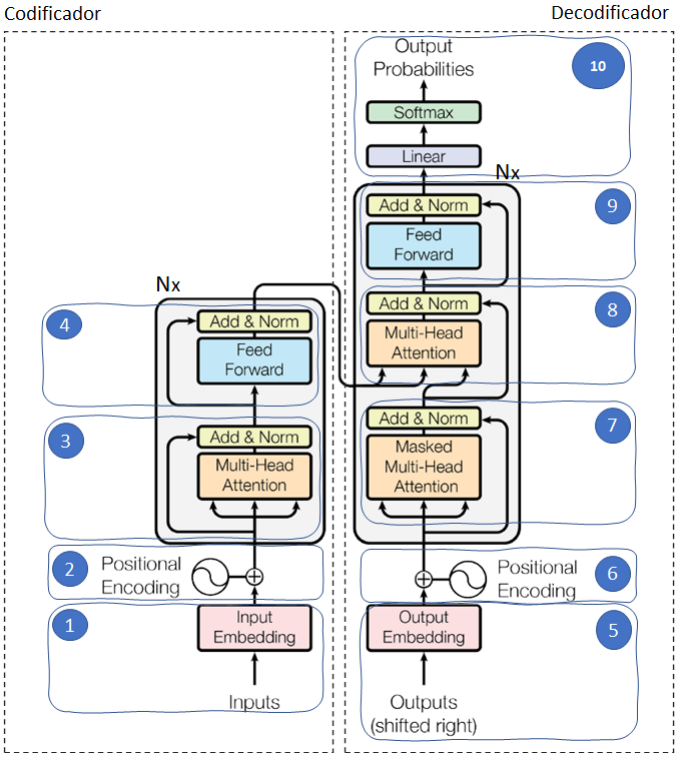

Figura 2.5. Arquitetura Transformer. Fonte: Adaptado de [Vaswani et al., 2017].

A arquitetura apresentada na Figura 2.5 recebe como entrada, na parte inicial do codificador, uma frase inteira, e fornece como saída, na parte final do decodificador a probabilidade das próximas palavras a serem preditas.

Cabe ressaltar que as saídas do decodificador também são utilizadas para retroalimentar o processo, funcionando também como entrada na parte inicial do decodificador. Dessa forma, o decodificador também recebe como entrada uma frase inteira. Ou seja, a frase é passada para o codificador, faz-se uma série de processamentos e depois vai para o decodificador. Então, o decodificador faz as previsões, que retornam as probabilidades de cada uma das palavras. Essas probabilidades são utilizadas para realimentar a entrada do decodificador. Ou seja, as palavras previstas são utilizadas como entrada para novas previsões.

As próximas subseções detalham o funcionamento de cada um dos componentes da arquitetura *transformer*.

2.5.1. *Input Embedding*

A subcamada *Input Embedding* é a camada de entrada do codificador e tem a função de receber as palavras de entrada e convertê-las em vetores. Essa subcamada é representada pela estrutura marcada com o número 1 na Figura 2.5.

2.5.2. *Positional Encoding*

O *Positional Encoding* é utilizado para marcar a posição das palavras dentro das frases. As arquiteturas baseadas em convoluções (CNNs) ou em recorrências (RNNs) não precisam desse tipo de recurso. No caso das convoluções, as palavras sequenciais ficam juntas no mesmo filtro, ou seja, o posicionamento das palavras é mantido. As RNNs também mantêm a ordem das palavras, de acordo com a ordem de entrada dessas palavras. Sendo assim, essas duas arquiteturas permitem a manutenção da ordem sequencial das palavras.

Já na arquitetura *transformer*, faz-se necessário um marcador de posição, que aplica uma fórmula matemática, na matriz de *embbeding*, para que seja possível a identificação da posição de cada palavra no texto.

Vaswani et al. (2017) resolvem essa questão utilizando as funções seno e cosseno para gerar diferentes frequências para a codificação posicional (PE). O *Positional Encoding*, representado pela estrutura de número 2 na arquitetura apresentada na Figura 2.5, cria outra matriz de *embbeding* adicionando um valor posicional, que marca a ordem das palavras. As Equações 6 e 7 indicam como se dá a obtenção do *positional encoding*.

$$PE_{(pos,\ 2i)} = \sin\left(\frac{Pos}{100000^{2i/d_{model}}}\right) \qquad [Vaswaniet\ al.,\ 2017] \tag{6}$$

$$PE_{(pos,\ 2i+1)} = \cos\left(\frac{Pos}{100000^{2i/d_{model}}}\right) \qquad [Vaswaniet\ al.,\ 2017] \tag{7}$$

Como pode ser observado, para as posições pares $(2i)$ utiliza-se a função seno, e para as posições ímpares $(2i+1)$ utiliza-se a função cosseno.

Logo, a função dessa subcamada é adicionar um valor de codificação posicional ao *embedding* de entrada, a fim de possibilitar a recuperação da posição da palavra dentro da frase.

2.5.3. *Multi-Head Attention*

As subcamadas *Multi-Head Attention*, representadas pelos números 3 e 8 na Figura 2.5, recebem como entrada um vetor de embedding que é dividido em três fluxos iguais. Logo, essa camada recebe três entradas, sendo que essas entradas são idênticas e representam os vetores de palavras da frase de entrada.

Esse componente é responsável por implementar o mecanismo de atenção. O mecanismo de atenção é um recurso que permite que uma determinada região do texto receba um "foco" maior, fazendo com que as demais informações fiquem "desfocadas". Vaswani et al. (2017) definem esse tipo de mecanismo como uma função que realiza o mapeamento de uma *Query* (Q) e um conjunto de pares de chave-valor (K,V) para uma saída. A *Query* (Q), as chaves (K), os valores (V) e a saída são matrizes formadas pelos vetores das palavras da frase de entrada. A saída é calculada como uma soma ponderada dos valores, em que o peso atribuído a cada valor é calculado por uma função de compatibilidade da *Query* com a chave correspondente.

Essa atividade é executada por uma estrutura denominada *Scaled dot-product attention*, cuja funcionalidade é descrita abaixo

- ***Scaled dot-product attention***

O *scaled dot-product attention* trata a relação da frase original com suas próprias palavras. Sendo assim, esse mecanismo utiliza duas sentenças iguais A e B, e calcula como cada elemento de A está relacionado a cada elemento de B. Após isso, recombina-se A de acordo com essa relação. Ou seja, esse tipo de mecanismo identifica palavras relacionadas dentro de uma mesma frase.

Sendo assim, a partir de uma sequência A (com um contexto B), obtém-se uma nova sequência na qual cada elemento é uma mistura dos elementos de A que estão relacionados com B. Assim, tem-se uma matriz que faz a combinação da frase com ela mesma.

Dessa forma, um produto escalar indica a similaridade entre dois vetores (no caso duas palavras). Ou seja, considerando u e v dois vetores de *embeddings* que representam duas palavras em uma mesma frase, quanto mais distante u está de v, menor a similaridade entre essas palavras. Da mesma forma, quanto mais próximos estes vetores estiverem, maior é a similaridade entre as palavras. Logo, a partir do produto escalar, realizado para todos os pares de palavras da frase, tem-se a relação existente entre cada par de palavra dessa frase.

A Equação 8 representa a operação executada pelo *scaled dot-product attention*, sendo que, Q, K e V são iguais e representam a frase de entrada.

$$Attention(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad [Vaswaniet\ al.,\ 2017] \qquad (8)$$

A Figura 2.6 representa graficamente a Equação 8. Nesse mecanismo, primeiro faz-se a multiplicação de Q e K (QK^T, sendo que K^T representa a transposta da matriz K). No denominador da função softmax, d indica a dimensão do *embbeding* que representa os tokens. Essa divisão é utilizada para alterar a escala do resultado da multiplicação das matrizes. Posteriormente, aplica-se a função softmax (que fornece um valor entre 0 e 1). Por fim, faz-se a multiplicação pela matriz V.

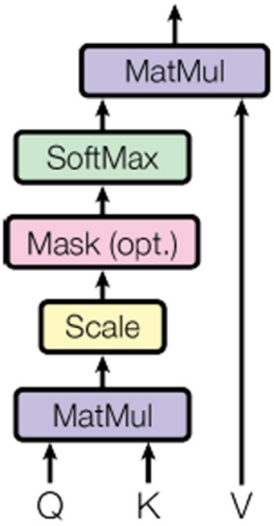

Figura 2.6. *Scaled dot-product attention*. Fonte: [Vaswani et al., 2017].

Na Equação 8, a multiplicação de Q por K indica a autocorrelação das palavras da frase (pois, Q e K representam uma mesma frase). Logo, essa multiplicação retorna uma matriz que indica como as palavras da frase estão relacionadas entre si. O retorno da multiplicação QK é uma matriz, que é multiplicada por V e origina uma nova matriz.

Para exemplificar a forma como o *scaled dot-product attention* funciona, utilizou-se frase: “eu gosto de café”. Considerando que cada uma das palavras está sendo representada por um vetor de 5 posições, os vetores das palavras podem ser ilustrados conforme os vetores apresentados na Figura 2.7(a). Já a frase completa é representada pela junção desses vetores, conforme apresentado na Figura 2.7(b). Ou seja, a frase será representada por uma matriz com 4 linhas (cada uma referente a uma das palavras da frase) e com 5 colunas (cada coluna referente a uma das dimensões utilizadas para representar as palavras). Cabe ressaltar que as 3 entradas do *scaled dot-product attention* são cópias dessa matriz que representa a frase inteira e essas entradas são chamadas de Q, K e V.

A primeira parte do *scaled dot-product attention* faz a multiplicação entre as matrizes Q e K^T. Nessa multiplicação, cada linha da matriz Q é associada com uma coluna da matriz K^T. Ou seja, cada vetor que representa uma palavra na matriz Q é multiplicado

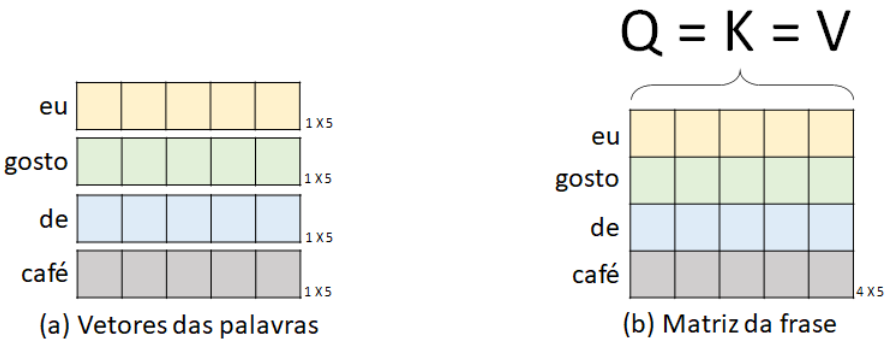

Figura 2.7. Representação da Frase. Fonte: Elaborada pelos autores.

por um vetor que representa uma palavra na matriz K^T, sendo que, o resultado dessa multiplicação expressa o grau de associação de cada uma das palavras com todas as demais palavras que compõem a frase de entrada. Dessa forma, ao final tem-se como resultado uma matriz quadrada, cuja dimensão coincide com o número de palavras da frase. Essa matriz de saída fornece o auto relacionamento das palavras de uma mesma frase. Assim, torna-se possível identificar palavras que possuem ligações maiores e sentidos semelhantes. A Figura 2.8 ilustra esse processo.

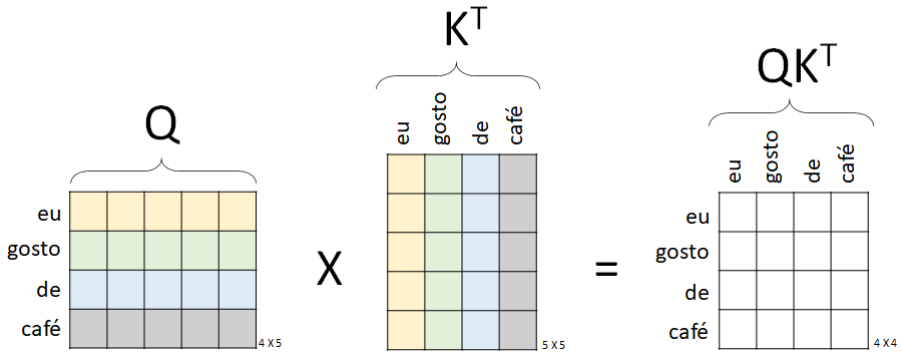

Figura 2.8. Cálculo do relacionamento entre as palavras. Fonte: Elaborada pelos autores.

O resultado da multiplicação é dividido pela raiz da dimensão e aplica-se a função $softmax$, fazendo assim, que todos os valores fiquem compreendidos entre 0 e 1. O passo final é a multiplicação do resultado da função $softmax$ com a matriz V. Ou seja, a saída do *scaled dot-product attention* é uma matriz com as mesmas dimensões da matriz de entrada. A Figura 2.9 ilustra esse processo.

- **Projeções Lineares no *Multi-Head attention Layer***

Vaswani et al. (2017) identificaram que, ao invés de executarem uma única função de atenção com todo o texto de entrada, seria melhor trabalhar com projeções lineares dessa entrada. Sendo assim foi proposta a utilização de um número h de projeções (que são aplicadas nos textos de entrada), conforme demonstrado na Figura 2.10.

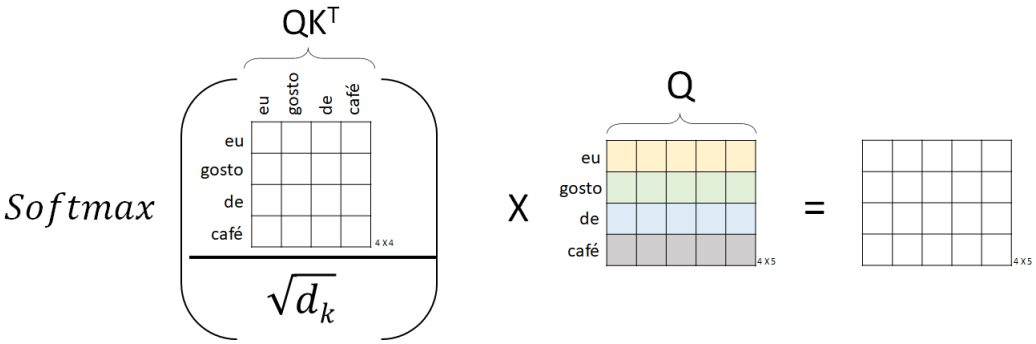

Figura 2.9. Saída do *scaled dot-product attention*. Fonte: Elaborada pelos autores.

O conceito de projeções lineares, permite a aplicação de mecanismos de atenção em diferentes subespaços. Ou seja, diferentes projeções lineares possibilitam o processamento de informações de diferentes subespaços em diferentes posições.

Dessa forma, antes de se aplicar o produto escalar, realiza-se a divisão das entradas em subespaços. O objetivo dessa divisão é fazer com que a relação entre as palavras fique mais forte. Isso ocorre pelo fato do produto escalar ser aplicado em um conjunto menor de palavras. Logo, quando se tem palavras relacionadas, o cálculo de similaridade entre as palavras retorna um valor maior, deixando mais evidente a relação entre as palavras.

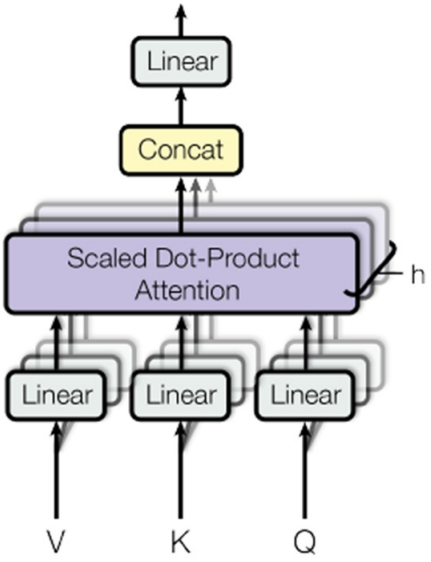

Figura 2.10. Projeções Lineares no *Multi-Head attention Layer*. Fonte: [Vaswani et al., 2017].

Caso essas projeções não fossem realizadas, a relação entre as palavras seria mais fraca, pois mais palavras seriam levadas em consideração durante o cálculo de similaridade.

2.5.4. *Feedforward Network*

As camadas *Feed Forward*, representadas pelo número 4 na Figura 2.5, são responsáveis pela aplicação de duas transformações lineares (duas camadas densas), conforme pode ser observado na Equação 9.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \qquad [Vaswaniet\ al.,\ 2017] \tag{9}$$

2.5.5. Camada de Normalização (*Add & Norm*)

Cada subcamada de atenção e cada subcamada *Feed Forward* do *transformer* é seguida por uma subcamada de normalização (*Add & Norm*). Essas subcamadas aparecem após várias estruturas da arquitetura *transformers*, como pode ser observado nas estruturas representadas pelos números 3, 4, 7, 8 e 9 na Figura 2.5.

O objetivo da camada *Add & Norm* é evitar que as informações da camada anterior sejam esquecidas, o que ajuda durante o processo de *backpropagation*.

Como pode ser observado na arquitetura, a camada *Add & Norm* sempre recebe uma cópia das frases, antes delas entrarem nos mecanismos de atenção (ou nas camadas *Feed Forward*). Isso permite que, além dos valores processados, os valores originais também sejam mantidos. Dessa forma, tem-se a versão original e a versão transformada dos textos.

Essa camada de normalização contém uma função de adição e um processo de normalização de camada. A função *Add* processa as conexões residuais que vêm da entrada da subcamada. Essa operação garante que informações críticas não sejam perdidas. A Equação 10 descreve o funcionamento dessa camada, sendo que, x representa o texto original e $sublayer(x)$ representa a saída da camada imediatamente anterior à camada de normalização (uma camada de atenção ou *Feed Forward*).

$$LayerNorm(x + Sublayer(x)) \qquad [Vaswaniet\ al.,\ 2017] \tag{10}$$

2.5.6. *Masked Multi-head Attention*

O *Masked Multi-head Attention* é uma estrutura que só está presente no decodificador. Essa estrutura funciona de maneira similar ao *Multi-head Attention*, a única diferença é que ela aplica uma máscara na matriz de entrada. A função dessa máscara é mascarar as palavras localizadas após a palavra que está sendo tratada. Dessa forma, a arquitetura aprende a prever essas palavras futuras que foram mascaradas.

Cabe ressaltar que a subcamada *Masked Multi-head Attention*, representada pelo número 7 na Figura 2.5, também é seguida por uma subcamada de normalização, que funciona de forma idêntica à descrita na Subseção 2.5.5.

2.5.7. Última camada Linear e Camada Softmax

Na saída do decodificador há uma última camada densa, essa camada tem o tamanho do vocabulário. Finalmente, o último componente da arquitetura é uma função softmax, que

é aplicada com o objetivo de gerar as probabilidades de cada uma das palavras. Essas estruturas são representadas pelo número 10 na Figura 2.5.

2.6. *Bidirectional Encoder Representations from Transformers* (BERT)

BERT (*Bidirectional Encoder Representations from Transformers*) [Devlin et al., 2019] é um modelo de representação de linguagem baseado na arquitetura *transformer* [Vaswani et al., 2017]. Esse modelo é considerado bidirecional pelo fato das palavras serem representadas com base tanto no contexto à esquerda quanto no contexto à direita. Tais representações são obtidas a partir de treinamentos em bases de dados textuais não rotuladas.

A partir do modelo do BERT, surgiu uma série de modelos que apresentam algumas variantes em relação ao modelo original: ROBERTA [Liu et al., 2019], ALBERT [Lan et al., 2020], DistilBERT [Sanh et al., 2019] e etc. No entanto, este estudo ficará restrito ao BERT.

2.6.1. Modelos Anteriores

Durante o processo de evolução das técnicas de processamento de linguagem natural, muitos estudos apresentaram resultados satisfatórios, inspirando assim o surgimento de novos modelos. Dentre os modelos que precederam o BERT pode-se destacar o Elmo [Peters et al., 2018] e o OpenAI GPT [Radfort et al., 2018].

- **ELMO**

O ELMO [Peters et al., 2018] não implementa o conceito de *transformers*, nem de transferência de aprendizagem. Esse modelo apresenta duas características principais: é considerado pseudo bidirecional e utiliza redes neurais recorrentes.

A arquitetura ELMO é classificada como pseudo bidirecional porque, apesar de considerar tanto o contexto à esquerda quanto o contexto à direita das palavras, ela não considera a sentença inteira, de uma única vez. A sentença não passa por um núcleo único, que processa todo o texto de forma conjunta. Sendo assim, a sentença é dividida em duas partes: uma à esquerda da palavra que está sendo considerada e outra à direita dessa palavra.

A Figura 2.11 ilustra a arquitetura ELMO, sendo que, cada elemento E_i representa uma frase de entrada. Como pode ser observado, cada frase é repassada tanto para a parte esquerda quanto para a parte direita da arquitetura. Ao final do processo, os resultados desses dois núcleos de processamento são concatenados em uma única saída.

Cabe ressaltar que as setas internas nos dois núcleos de processamento, apresentados na Figura 2.11, representam a direção em que as informações são processadas.

O fato dessa arquitetura utilizar, células recorrentes (mais especificamente células LSTM) representa uma desvantagem para esse tipo de processamento. Essa desvantagem ocorre porque as LSTMs fazem sucessivas chamadas recursivas, e à medida que as camadas ficam mais profundas a informação tende a ser perdida.

- **OpenAI GPT**

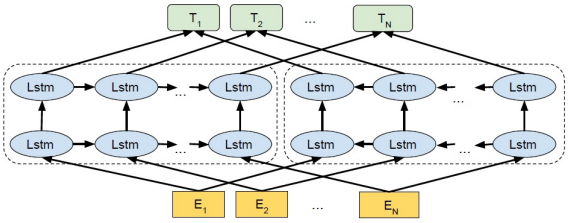

Figura 2.11. Arquitetura ELMO. Fonte: [Devlin et al., 2019].

A arquitetura OPEN AI GPT [Radfort et al., 2018], apresentada na Figura 2.12, é baseada em *transformers*. No entanto, ela só considera o contexto à esquerda dos *tokens*. Sendo assim, a principal deficiência desse modelo é que ele não implementa o conceito bidirecional.

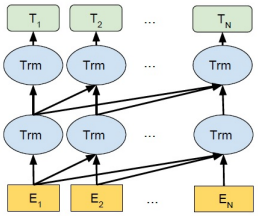

Figura 2.12. Arquitetura GPT-1. Fonte: [Devlin et al., 2019].

2.6.2. Arquitetura BERT

O BERT [Devlin et al., 2019] combina as características dos modelos anteriores: utiliza a arquitetura transformer e é bidirecional. A bidirecionalidade é obtida pelo fato das frases serem processadas tanto em relação aos dados à esquerda quanto à direita. Dessa forma, durante o treinamento, uma frase é passada tanto para a direita quanto para a esquerda, porém em um mesmo centro de processamento.

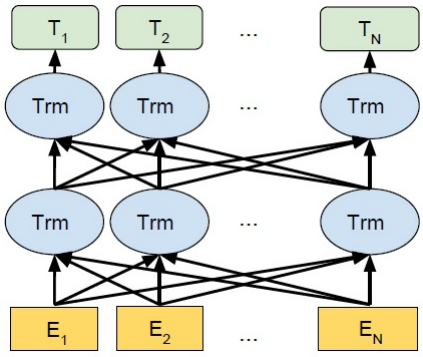

Figura 2.13. Arquitetura BERT. Fonte: [Devlin et al., 2019].

O BERT é composto por uma série de codificadores (*transformers*) empilhados. Dessa forma, o BERT utiliza apenas parte da arquitetura *transformer*. Esses codificadores empilhados permitem que as sentenças sejam codificadas de maneira mais efetiva, possi-

bilitando uma melhor composição dessas sentenças, de acordo com as relações entre as palavras.

O BERT possui duas arquiteturas possíveis, o *BERT BASE* e o *BERT LARGE*, sendo que a principal diferença entre essas arquiteturas é o número de codificadores empilhados. O *BERT BASE* é constituído por uma sequência de 12 codificadores, enquanto o *BERT LARGE* é composto por 24 codificadores empilhadas.

Apesar dessa diferença principal, as variantes do BERT podem ser definidas pela combinação de 3 parâmetros: *L*, *H* e *A*. O parâmetro *L* indica o número de camadas de codificadores. Já o parâmetro *H* indica a dimensão dos *embbedings* gerados (número de unidades na última camada oculta da arquitetura) e o parâmetro *A* indica o número de *self-attentions heads*. A Figura 2.14 apresenta a composição dos modelos *Base* e *Large* do BERT. Cabe ressaltar que o modelo *Base* possui cerca de 110 milhões de parâmetros, enquanto o modelo *Large* 340 milhões.

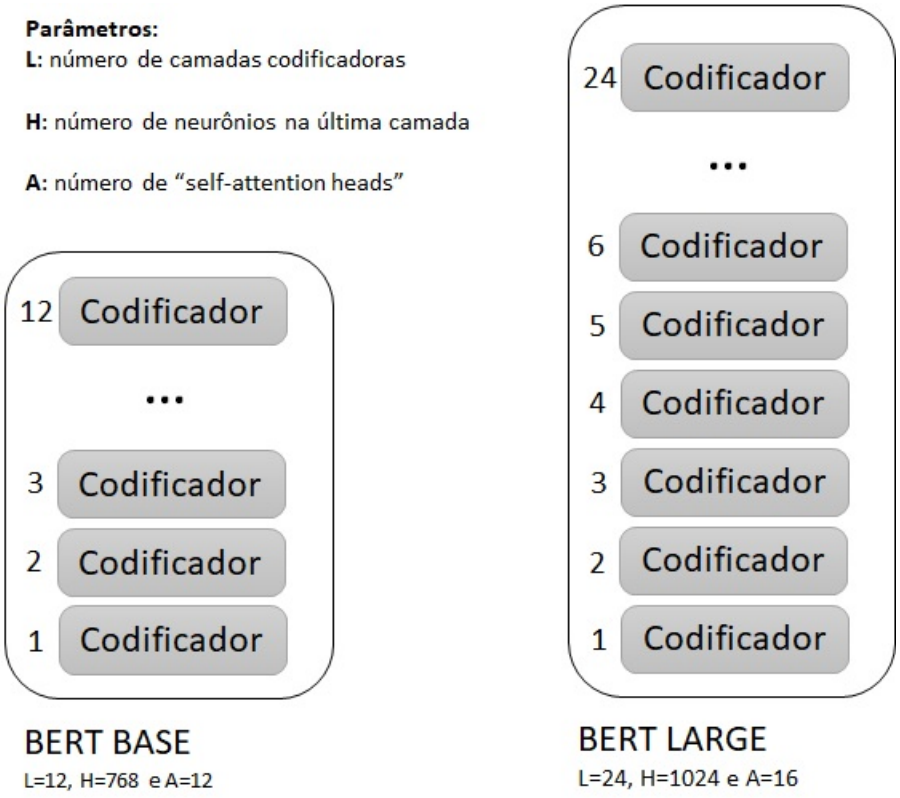

Figura 2.14. composição dos modelos Bert. Fonte: Elaborada pelos autores.

2.6.3. Treinamento do BERT

O modelo original do BERT foi pré-treinado utilizando o *BookCorpus* (corpus com textos de livros com cerca de 800 milhões de palavras) e a Wikipedia em inglês (com cerca de 2,5 milhões de palavras). O pré-treinamento utilizou 16 TPUs[8] e demorou cerca de 3

[8]Tensor Processing Unit-TPU é uma unidade de processamento de tensores desenvolvidas pelo Google para acelerar o processamento em aplicações de deep learning.

dias. Ou seja, esse processo é complexo e exige um grande poder computacional para executá-lo.

O pré-treinamento do BERT é composto por duas tarefas: o *Masked Language Modeling* (MLM) e o *Next Sentence Prediction* (NSP).

- ***Masked Language Modeling* (MLM)**

Na tarefa *Masked Language Modeling*, realiza-se um mascaramento aleatório de uma percentagem do *tokens* de entrada, para que o modelo possa aprender como predizê-los. Dessa forma, substitui-se a palavra a ser predita, colocando-se o *token [MASK]* no lugar, e tenta-se obter o *embedding* desse *token* na saída, conforme ilustrado na Figura 2.15.

Devlin et al. (2019) deixam claro que, diferentemente dos auto-encoders de eliminação de ruído Vincent et al. [2008], o objetivo dessa tarefa não é reconstruir toda a sequência de palavras de entrada, mas apenas fazer a previsão da palavra mascarada.

Cabe ressaltar que, esse token $[MASK]$ não é empregado em todas as iterações, ele é utilizado em apenas 15% das palavras. Ou seja, apenas 15% das palavras são substituídas, sendo que, tal substituição obedece a seguinte regra: em 80% dos casos substitui-se pelo token $[MASK]$, em 10% dos casos substitui-se por um token aleatório, e em 10% dos casos, não se modifica o token.

Devlin et al. (2019) informam que a vantagem desse procedimento é que o codificador do *transformer* não sabe quais palavras ele terá que prever, e nem quais foram substituídas por palavras aleatórias, então ele é forçado a manter uma representação contextual distributiva de cada *token* de entrada. Além disso, como a substituição aleatória ocorre em uma pequena percentagem dos *tokens*, isso não prejudica a capacidade do modelo compreender a linguagem.

A Figura 2.15 ilustra a tarefa de treinamento denominada MLM. Nessa Figura, utiliza-se uma frase como entrada e aplica-se um mascaramento aleatório. Sendo assim, no exemplo em questão, a palavra correr foi substituída pelo token $[MASK]$, logo, deseja-se fazer a previsão dessa palavra. Ao final do processo aplica-se uma camada totalmente conectada (*feed forward neural network*) seguida de uma função softmax, para gerar a probabilidade para cada uma das classes (possíveis *tokens*). Logo, tem-se um problema de classificação em que se retorna uma probabilidade para cada uma dos *tokens* do vocabulário, sendo que, o objetivo é que o *token* mascarado (correr) seja o *token* com maior probabilidade.

- ***Next Sentence Prediction* (NSP)**

A segunda tarefa de treinamento do BERT é a Predição da Próxima Sentença. A entrada consiste em duas frases e o modelo deve dizer se a segunda sentença é sequência da primeira ou não.

Dessa forma, essa tarefa consiste em uma classificação binária, sendo que, sempre que a segunda sentença é sequência da primeira o rótulo é positivo, e no caso contrário o rótulo é negativo.

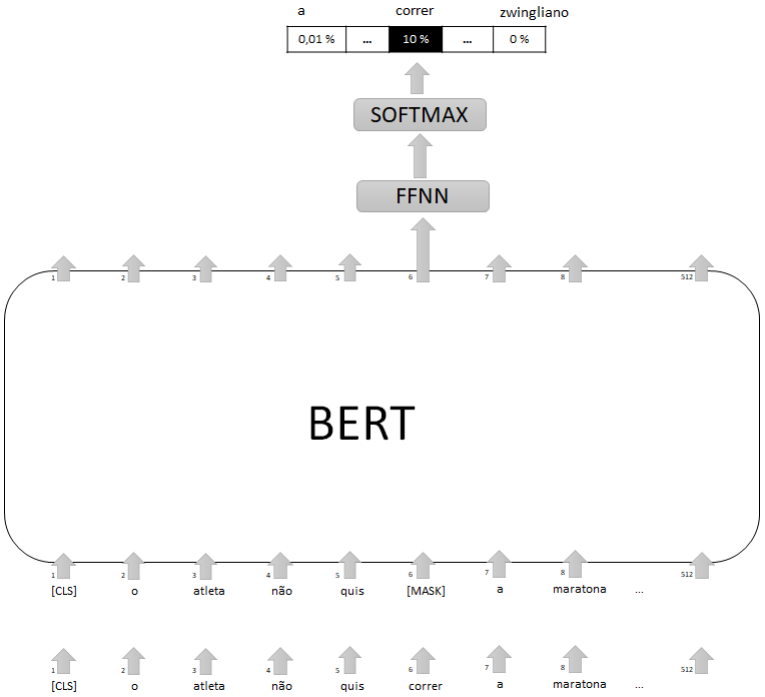

Figura 2.15. Treinamento "Masked Language Modeling". Fonte: Elaborada pelos autores.

A tarefa utiliza dois tokens especiais: $[CLS]$ e $[SEP]$. O $[CLS]$ é um token de classificação binário que é adicionado ao início da primeira frase para prever se a segunda frase é sequência da primeira. Já o $[SEP]$ é um token de separação que sinaliza o fim de uma frase. Sendo assim, a primeira frase é separada da segunda pelo token $[SEP]$.

As duas tarefas (MLM e NSP) são executadas simultaneamente, sendo que, as sequências de entrada devem ser previamente preparadas. A Figura 2.16 ilustra esse preparo.

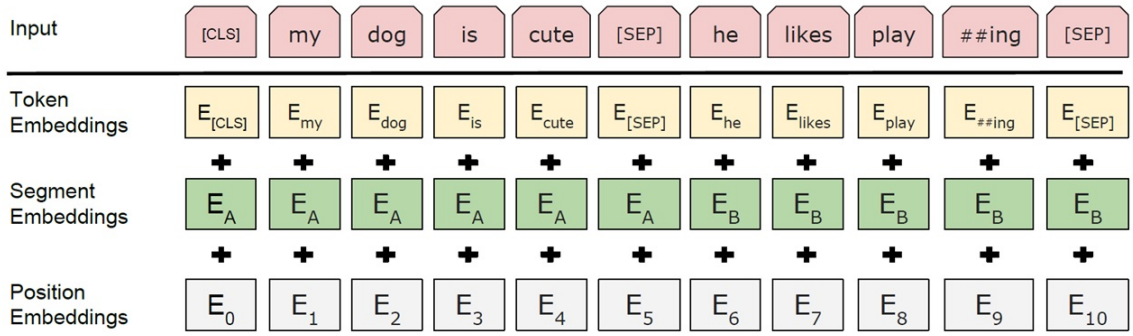

Figura 2.16. Entrada do modelo BERT. Fonte: [Devlin et al., 2019].

Conforme pode ser observado, os *embeddings* de entrada são obtidos pela soma dos *tokens embeddings* com os *segment embeddings* e os *embeddings* de codificação posicional. Essas operações podem ser resumidas da seguinte forma:

1. A sequência de palavras é dividida em *tokens*,

2. Substitui-se aleatoriamente alguns tokens pelo token $[MASK]$.

3. Um *token* de classificação $[CLS]$ é inserido no início de uma das sentenças e um *token* $[SEP]$ separa a primeira sentença da segunda.

4. O *embedding* das sentenças é adicionado ao *embedding* de *tokens*, de modo que a primeira frase tenha um valor de *embedding* diferente da segunda frase.

5. A codificação posicional é aprendida, sendo que, nesse caso não se utiliza a codificação posicional seno-cosseno do *transformer* original.

Na tarefa de NSP, busca-se um entendimento de nível superior, passando do nível de palavras para o nível de frases. Dessa forma, o modelo torna-se adequado para outros tipos de tarefas como por exemplo perguntas e respostas. A Figura 2.17 ilustra esse treinamento.

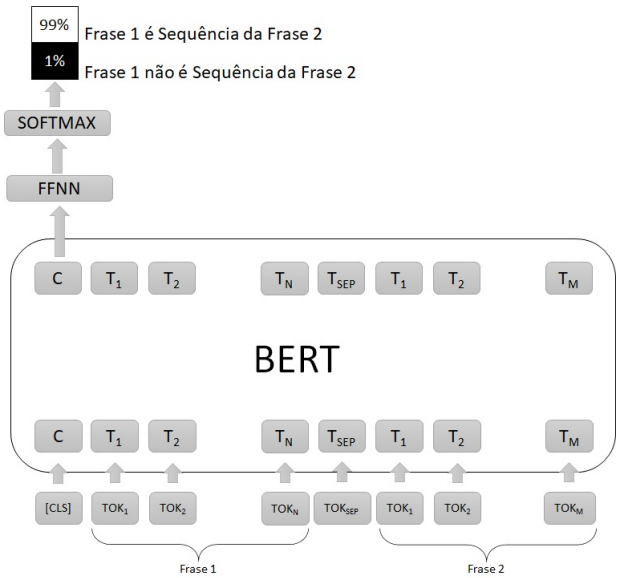

Figura 2.17. Treinamento "Next Sentence Prediction". Fonte: Elaborada pelos autores.

2.6.4. *Fine-tuning*

A grande vantagem do BERT é a possibilidade de transferência de aprendizagem. Dessa forma, uma vez que se tenha o modelo pré-treinado, esse pode ser utilizado em diversas tarefas de PLN, como por exemplo: classificação textual, verificação de semelhança entre sentenças, sistemas de perguntas e respostas e etc.

Para isso, deve-se realizar uma tarefa de *fine-tuning*, a fim de adaptar o modelo original a uma tarefa específica. Nesse sentido, Sun et al. (2019) investigam como maximizar a utilização do BERT para a tarefa de classificação de textos.

O modelo BERT recebe como entrada sequências de 512 *tokens* e fornece como saída vetores de 768 ou 1024 posições[9]. Sendo assim, a tarefa de *fine-tuning* consiste na inclusão de novas camadas ao modelo, a fim de adaptar a estrutura do modelo original ao problema em questão. Durante o *fine-tuning* também é realizado o treinamento com os dados específicos do problema em questão.

Sun et al. (2019) defendem que, quando se adapta o BERT para uma tarefa de PLN em um domínio específico, deve-se buscar uma estratégia de *fine-tuning* apropriada. Dessa forma, os autores sugerem 3 métodos de *fine-tuning*.

- ***Fine-tuning Strategies*:** quando se ajusta o BERT para uma tarefa específica, existem muitas formas de utilizar o BERT. Sun et al. (2019) argumentam que as diferentes camadas do BERT capturam diferentes níveis de informações semânticas e sintáticas. Dessa forma, torna-se necessário saber qual camada é a melhor para uma determinada tarefa. Os autores defendem ainda que, além da definição da camada, deve-se também escolher de forma adequada a taxa de aprendizagem e a estratégia de otimização.
- **Pré-treinamento adicional:** o BERT é treinado em textos de domínio geral (genéricos), com distribuições de dados diferentes da distribuição dos dados dos textos da tarefa de domínio específico, que se deseja adaptar. Sendo assim, Sun et al. (2019) sugerem um pré-treinamento adicional, com os dados referentes ao domínio da tarefa específica.
- ***Multi-task fine-tuning*:** Sun et al. (2019) defendem que, quando há várias tarefas disponíveis em um domínio específico, uma questão interessante é verificar se há benefícios no *fine-tuning* do BERT em todas as tarefas simultaneamente.

Sun et al. (2019) citam a limitação do BERT de trabalhar com tamanhos máximos de textos de até 512 tokens e sugerem dois métodos para solucionar essa dificuldade: o método do truncamento e o método hierárquico.

O método do truncamento consiste em truncar o texto de forma que esse passe a ter apenas 510 tokens (reserva-se dois tokens, para indicar o início e o final do texto). Para esse truncamento, Sun et al. (2019) citam 3 estratégias distintas: manter os 510 tokens iniciais, manter os 510 tokens finais ou seleciona os 128 primeiros tokens e os 382 tokens finais.

Nos métodos hierárquicos, Sun et al. (2019) propõem a divisão do texto de entrada em grupos de 510 tokens. Dessa forma, torna-se possível apresentar ao modelo cada um desses grupos de forma separada. Assim, obtém-se a representação de cada um desses grupos (de 510 tokens) e em seguida combina-se essas representações a fim de se obter uma representação final.

Sun et al. (2019) também citam o problema do esquecimento catastrófico, apresentado por McCloskey and Cohen (1989) . Esse tipo de problema ocorre em situações

[9] O modelo *BERT BASE* possui 768 neurônios na última camada oculta, e por tal razão a sua saída é um vetor de 768 valores. Já o *BERT LARGE*, apresenta com saída um vetor de 1024 posições, visto que, ele possui 1024 neurônios na última camada oculta.

de transferência de aprendizado, em que o conhecimento pré-treinado é apagado durante a aprendizagem de novos conhecimentos.

2.7. Outros Modelos

Um dos grandes avanços introduzidos pela arquitetura *transformers* foi a proposta de um mecanismo de atenção, que pode ser avaliado em paralelo para cada *token* da sequência de entrada, eliminando assim a dependência sequencial que ocorria em redes neurais recorrentes, como a LSTM.

No entanto, a limitação desse mecanismo é seu alto custo computacional. O fato de todos os *tokens* se relacionarem com todos os demais *tokens* da sequência torna quadrática a complexidade desse mecanismo (complexidade $O(n^2)$) .

Por tal razão, modelos derivados da arquitetura *transformers* possuem tamanhos máximos de entrada bem restritos. Um exemplo disso é o modelo BERT, visto anteriormente, que só trabalha com textos de até 512 *tokens*.

Sendo assim, passou-se a pesquisar formas de se alcançar os benefícios trazidos pelos mecanismos de atenção, introduzidos pela arquitetura *transformers*, sem o alto custo computacional.

Nesse sentido, foram propostas algumas soluções que diminuíam a complexidade do mecanismo de atenção. Dentre essas propostas estão as arquiteturas *Longformer* [Beltagy et al., 2020] e *Big Bird* [Zaheer et al., 2020], que possuem complexidade linear e por isso aceitam textos maiores.

2.7.1. *Longformer: The Long-Document Transformer*

Beltagy et al. (2020) definem alguns padrões de atenção que possibilitam a matriz de atenção escalar linearmente com o tamanho da sequência de entrada. Os padrões de atenção propostos foram a janela deslizante, a janela deslizante dilatada e a atenção global, que são apresentados na Figura 2.18.

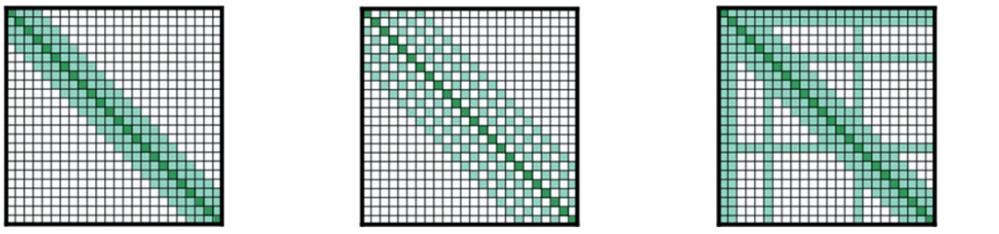

Figura 2.18. Padrões de atenção da arquitetura LongFormer. Fonte: Adaptado de [Beltagy et al., 2020]

A janela deslizante é o padrão que considera um tamanho de janela arbitrário w, e cada token se relaciona apenas com os w tokens mais próximos dele ($w/2$ à esquerda e $w/2$ à direita). Essa modificação faz com que a complexidade do mecanismo de atenção caia para $O(n)$. Ou seja, passa-se de uma complexidade quadrática para uma complexidade linear.

Os autores defendem a utilização de diferentes valores de w para camadas distintas, pois isso pode propiciar um equilíbrio entre a eficiência e a capacidade de representação do modelo.

Beltagy et al. (2020) também propõem a janela deslizante dilatada, com o objetivo de aumentar o campo receptivo sem aumentar a computação. Essa janela funciona de forma análoga a citada anteriormente, porém, com lacunas de dilatação.

Dessa forma, torna-se possível atingir um número maior de tokens, sem prejudicar a complexidade do algoritmo. Isso ocorre porque as lacunas deixam os campos respectivos mais amplos, mantendo o número de operações matemáticas.

Apesar das soluções anteriores resolverem o problema da complexidade do mecanismo de atenção, elas não têm condições de tratar dependências de longa duração. Ou seja, elas não permitem a identificação de relacionamentos entre *tokens* que estejam muito distantes no texto.

Sendo assim, para tratar esse tipo de situação, os autores propõem também a utilização do mecanismo de atenção tradicional. No entanto, a ideia não é calcular a relação entre todos os *tokens*, como acontece em [Vaswani et al., 2017]. Nesse caso, apenas alguns *tokens* pré-selecionados tem a sua atenção totalmente calculada.

Os autores ressaltam que essa operação de atenção é simétrica: ou seja, um *token* com uma atenção global atende a todos os *tokens* na sequência e todos os *tokens* na sequência atendem a ele.

Porém, como o número desses *tokens* selecionados não depende do tamanho da sequência (ou seja, a quantidade de *tokens* que recebem a atenção global não varia com o tamanho de n), a complexidade desse mecanismo de atenção continua sendo $O(n)$.

Sendo assim, o *Longformer* trabalha com mecanismos de atenção que combinam janelas deslizantes com a aplicação de atenção global para alguns *tokens* selecionados. Essa combinação consegue resultados semelhantes aos dos *transformers* originais, porém, com complexidade bem menor.

2.7.2. *Big Bird: Transformers for Longer Sequences*

Zaheer et al. (2020) também propõem um mecanismo de atenção esparsa que reduz a complexidade quadrática dos mecanismos originais para uma complexidade linear. Esse modelo pode ser entendido como uma combinação de 3 tipos de mecanismos de atenção: atenção aleatória, janela de atenção e atenção global. A Figura 2.19 ilustra o funcionamento desses mecanismos.

A atenção aleatória é uma proposta em que um determinado *token* só se conecta com alguns *tokens* aleatoriamente selecionados, ao invés de se conectar com todos os demais *tokens* da sequência, como ocorre no mecanismo de atenção original.

Sendo assim, supondo que sejam selecionados r *tokens* para se conectarem com um determinado *token* (por exemplo $r = 2$), a complexidade deixa de ser $O(n^2)$ e passa a ser $O(r * n)$. Porém, como r é um valor constante, pode-se dizer que a complexidade é $O(n)$.

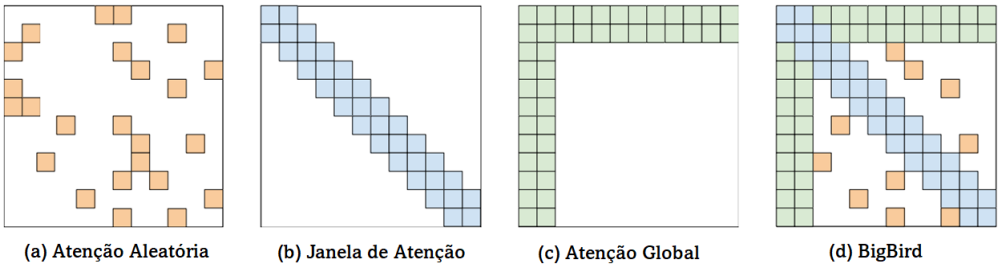

Figura 2.19. Mecanismo de atenção do BigBird. Fonte: Adaptado de [Zaheer et al., 2020]

Ou seja, para cada *token* da sequência seleciona-se um número aleatório de *tokens* com quem ele irá se relacionar, sendo que esse número aleatório é fixo, não dependendo assim do tamanho da sequência.

Já a janela aleatória funciona de maneira análoga à janela deslizante proposta por Beltagy et al. [2020]. Nesse tipo de mecanismo, cada *token* se comunica com um número fixo w de vizinhos mais próximos. Logo, a complexidade passa de $O(n^2)$ para $O(w * n)$. No entanto, como w também é fixo, pode se dizer que a complexidade é $O(n)$.

Por fim, na atenção global, seleciona-se alguns *tokens* que se conectam a todos os demais *tokens*. Ou seja, essa estrutura também é similar à empregada em [Beltagy et al., 2020].

O Big Bird une essas três estruturas a fim de propor uma aproximação para o mecanismo de atenção original. Ou seja, busca-se resultados semelhantes com menos complexidade. Essas alterações permitem a utilização de sequências de textos maiores.

2.8. Conclusão

Esse Capítulo apresentou um estudo sobre as principais técnicas de deep learning para o processamento de linguagem natural. No entanto, esse assunto é muito amplo e constantemente novas técnicas estão sendo propostas. Dessa forma, esse trabalho não teve a pretensão de esgotar esse assunto, sendo apenas um ponto de partida para estudos mais profundos nessa área.

Referências

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *ArXiv*, abs/2004.05150.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chollet, F. (2017). *Deep learning with Python*. Simon and Schuster, Shelter Island, NY 11964, 1 edition.

Devlin, J., Chang, M. W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1(Mlm):4171–4186.

Díaz-Guerra, L., Daher Adegas, F., Stoustrup, J., and Monros, M. (2012). Adaptive control algorithm for improving power capture of wind turbines in turbulent winds. In *2012 American Control Conference (ACC)*, pages 5807–5812.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Goyal, P., Pandey, S., and Jain, K. (2018). *Deep Learning for Natural Language Processing*. Apress, Berkeley, CA.

Han, D., Liu, Q., and Fan, W. (2018). A new image classification method using CNN transfer learning and web data augmentation. *Expert Systems with Applications*, 95:43–56.

Harris, Z. S. (1954). Distributional Structure. *WORD*, 10(2-3):146–162.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., et al. (2018). Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Johnson, R. and Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. In Mihalcea, R., Chai, J. Y., and Sarkar, A., editors, *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 103–112. The Association for Computational Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In Moschitti, A., Pang, B., and Daelemans, W., editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL.

Kosmajac, D. and Kešelj, V. (2019). Automatic text summarization of news articles in serbian language. In *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6. IEEE.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C):109–165.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*.

Paiva, E., Paim, A., and Ebecken, N. (2021). Convolutional neural networks and long short-term memory networks for textual classification of information access requests. *IEEE Latin America Transactions*, 19(5):826–833.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *30th International Conference on Machine Learning, ICML 2013*, number PART 3, pages 2347–2355. PMLR.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 1, pages 2227–2237, Stroudsburg, PA, USA. Association for Computational Linguistics.

Radfort, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). (OpenAI Transformer): Improving Language Understanding by Generative Pre-Training. *OpenAI*, pages 1–10.

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Sun, C., Qiu, X., Xu, Y., and Huang, X. (2019). How to Fine-Tune BERT for Text Classification? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11856 LNAI(2):194–206.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):5999–6009.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 1096–1103, New York, New York, USA. ACM Press.

Yadav, A. and Vishwakarma, D. K. (2020). Sentiment analysis using deep learning architectures: a review. *Artificial Intelligence Review*, 53(6):4335–4385.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020). Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297.

Zhang, X., Zhao, J. J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., and Xie, X. (2016). Co-occurrence Feature Learning for Skeleton based Action Recognition using Regularized Deep LSTM Networks. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 3697–3703.