Pós-Graduação em Inteligência Artificial Generativa e LLMs

PUC-Rio

Business Intelligence

Notas de Aula

Vítor Wilher

BI · Eletiva

Versão 1.0

Resumo. Inteligência de negócios: modelagem dimensional, ETL, indicadores e visualização analítica.
Palavras-chave: business intelligence; ETL; indicadores; dashboards

Notas de estudo elaboradas a partir do material das aulas.

Índice

1 Visão Geral da Disciplina

A disciplina de Business Intelligence apresenta uma introdução às principais técnicas de BI disponíveis, expondo os conceitos essenciais e as ferramentas utilizadas no mercado para cada etapa do processo. O fio condutor é a construção de um projeto de implantação de BI de ponta a ponta: parte-se do levantamento das necessidades do negócio, passa-se pela modelagem multidimensional, pela definição da arquitetura do Data Warehouse, pelo projeto e pela automação do ETL, e chega-se à camada de visualização de dados. Ao final do percurso tradicional de BI, a disciplina amplia o escopo para dados não estruturados, apresentando Processamento de Linguagem Natural, representações vetoriais de texto, LLMs, RAG e agentes de IA.

O planejamento original previsto na ementa contempla oito aulas expositivas, oito oficinas com exercícios e desafios sobre cada tema, e duas Talks sobre projetos de BI. As competências que se pretende desenvolver são: entender a motivação e os benefícios do BI; elaborar um projeto de Data Warehouse; praticar a abordagem de modelagem multidimensional; conhecer o processo de ETL; versionar e atualizar o processo de ETL; criar um pipeline de dados automatizando o ETL com ferramenta de mercado; analisar e visualizar dados do Data Warehouse com ferramentas de mercado; e criar um projeto prático de BI.

O framework tecnológico adotado na disciplina é explícito e coerente com o mercado: **PostgreSQL** como SGBD tanto para a Stage Area quanto para o Data Warehouse, **Power Architect** para a modelagem do modelo estrela, **Pentaho Data Integration (PDI)** para o ETL e os pipelines, e **Power BI** para os dashboards. As aulas se encadeiam de forma cumulativa: o modelo multidimensional definido na segunda aula guia o ETL das aulas seguintes; o ETL alimenta o DW; o DW alimenta os dashboards; e o job do PDI automatiza todo o ciclo.

A segunda metade da disciplina desloca o foco dos dados estruturados para os não estruturados. O argumento central é que o texto não estruturado é muito comum e, na prática, pode representar a maioria das informações disponíveis para uma organização — e-mails, críticas e opiniões de clientes, redes sociais, laudos, documentos técnicos, livros e resumos. A partir daí desenvolvem-se as etapas de pré-processamento textual, a representação numérica de palavras, as medidas de similaridade e, finalmente, as aplicações modernas baseadas em modelos de linguagem: LLMs, RAG, OCR e sistemas multiagentes, ilustradas por casos aplicados ao setor elétrico.

2 TALKS BI

Material de slides não disponível para esta aula.

A ementa prevê duas Talks sobre Projeto de BI, indicadas como Talk 01 e Talk 02. O material de apoio associado ao bloco de projeto inclui templates de trabalho: Template do Projeto, Template de Levantamento dos Requisitos do Negócio, Roteiro para Entrevista, além de referências de Bases de Dados Públicas e de um Gerador de Dados. Esses artefatos correspondem, no fluxo da disciplina, à fase inicial do ciclo de vida de um projeto de BI, tratada em profundidade na aula de Fundamentos.

3 Fundamentos de Business Intelligence e Tecnologia de Data Warehouse

3.1 A transformação do conhecimento

O ponto de partida conceitual é a **pirâmide DIKW** (também chamada WKID ou DICS), que descreve a cadeia dado, informação, conhecimento e sabedoria. O exemplo usado nos slides é meteorológico e didático:

- **Dado**: `umidade_relativa_do_ar` = 90%; `velocidade_do_vento` = 60 km/h. São medições brutas, sem interpretação.
- **Informação**: "tais índices indicam que vai chover muito". O dado ganha significado quando contextualizado.
- **Conhecimento**: "historicamente o bairro que preciso ir costuma alagar quando chove muito". Aqui entra a experiência acumulada e o histórico.
- **Sabedoria**: "vou adiar o meu compromisso". É a decisão, a ação orientada pelo conhecimento.

Essa progressão sintetiza o propósito do BI: não parar no dado, e sim caminhar até a decisão.

3.2 O que é (e o que não é) Business Intelligence

Duas definições são trazidas. Para Machado (2010), BI é um conjunto de tecnologias orientadas a disponibilizar informação e conhecimento em uma empresa. Para Tyson (1997), Business Intelligence é um processo que envolve a coleta, análise e validação de informações sobre concorrentes, clientes, fornecedores, candidatos potenciais à aquisição, candidatos a joint-venture e alianças estratégicas, incluindo eventos econômicos, regulatórios e políticos com impacto sobre os negócios; o processo analisa e valida essas informações e as transforma em conhecimento estratégico.

O material distingue dois tipos de análise. O **BI tradicional faz análise descritiva**: analisa o passado para tomar decisões no presente e se organizar para o futuro. Já a **análise preditiva** usa dados do passado para prever resultados futuros. Os níveis de maturidade em data analytics acrescentam ainda a dimensão prescritiva, que responde ao que fazer para que algo aconteça — ou para que não aconteça.

Igualmente importante é a delimitação negativa. O BI **não é** uma ferramenta, **não é** um gerador de relatórios e **não é** uma agenda. O BI é um conceito, é um processo de análise, coleta, transformação e disseminação das informações para a tomada de decisão. Os slides enfatizam que os dados precisam ser tratados como bem corporativo, que o BI deve apoiar o Planejamento Estratégico e que a tendência de mercado é a cultura **Data Driven**.

3.3 Problemas que motivam o BI, benefícios e vantagens

Entre os problemas típicos das organizações sem BI estruturado, listam-se: inconsistência nos conceitos utilizados; proliferação de planilhas, bancos de dados pessoais e interfaces de extração com pouca relação entre si; limitações quanto ao nível de detalhe e ao tempo para obtenção de informações; reduzida capacidade de simulação e análise; e custos e dependência crescentes para a formatação de relatórios.

Os benefícios apontados incluem antecipar mudanças no mercado e ações dos competidores, entender oscilações do mercado em relação à empresa, aprender com sucessos e falhas, conhecer melhor produtos e clientes, visualizar novos negócios, rever práticas e processos, além de integrar, consolidar e cruzar dados, possibilitando vantagem competitiva.

As vantagens operacionais são a dinamicidade na recuperação das informações, o cruzamento de dados de múltiplas fontes, a disponibilidade da informação em tempo real conforme as definições de tempo de atualização, a transformação da informação em conhecimento estratégico, a minimização de riscos e o uso de fatos em vez de achismos.

3.4 O processo e o ciclo de vida do BI

O processo de BI pode ser representado por um esquema simples que vai das fontes de dados para uma **Stage Area** e desta para o Data Warehouse e os Data Marts. Sobre esse esquema, os slides propõem um ciclo de vida do BI corporativo (Nascimento, 2019) com as seguintes etapas:

1. **Levantamento das necessidades do negócio**: entrevistas, conhecimento do negócio, identificação do problema, prospecção da solução e conhecimento dos dados.
2. **Planejamento da solução**: definição da arquitetura (on-premise, cloud server), ferramentas e especificidades do projeto.
3. **Modelagem Multidimensional**: definição dos componentes do modelo, das tabelas fato e dimensão, medidas, métricas, indicadores de performance e granularidade.
4. **Definição do processo de ETL**: coleta, conversão, uniformização, derivação, consolidação e agregação de dados.
5. **Criação do Data Warehouse** e sua população.
6. **Análise e Visualização de Dados**: criação de cubos, reports, dashboards e aplicação de algoritmos de mineração.
7. **Aceite por parte do cliente**.
8. **Melhoria contínua**: testar, manter, atender novas demandas e buscar melhoria do processo.

Quanto aos componentes, as fontes de dados incluem sistemas legados, arquivos externos (XML, XLS, TXT, CSV, MDB, web scraping), sistemas de gestão integrados (ERP), dados transacionais diversos e CRM. Os destinos são a Stage Area, os Data Warehouses e os Data Marts. Com os dados, elaboram-se dashboards, consultas por cubos OLAP, mineração de dados e análises preditivas.

3.5 Repositórios de dados e Arquitetura de Medalhão

Os slides diferenciam quatro tipos de repositório:

- **Data Warehouse**: dados estruturados.
- **Data Lake**: dados não estruturados e semiestruturados.
- **Data Lakehouse**: união dos dois mundos.
- **Data Mesh**: dados descentralizados.

O processo de qualidade de dados no DW e no Data Lake pode seguir a **Arquitetura de Medalhas** (Medallion Architecture), cujo objetivo é organizar o fluxo de transformação em camadas de modo que a qualidade e a confiabilidade dos dados aumentem a cada etapa:

- **Camada Bronze**: dados brutos, no formato original em que foram coletados (CSV, JSON, etc.), com qualidade e estrutura variáveis.
- **Camada Prata**: dados curados, que passaram por limpeza, validação e padronização, com desduplicação, filtragem e adição de metadados.
- **Camada Ouro**: dados de alto valor, transformados, agregados e otimizados para análise de negócios, prontos para consulta rápida e combinação com outras fontes.

3.6 Técnicas de levantamento de requisitos

Após a entrevista, os analistas de BI já têm uma ideia geral das necessidades e características (features) que o projeto deverá possuir; o resultado dessa investigação é a lista de requisitos. O material distingue três níveis, adaptados de Leffingwell e Widrig (2003):

- **Necessidades**: correspondem aos problemas do domínio, aquilo que os stakeholders desejam ver resolvido após a entrega. Exemplo: o projeto deve permitir entender melhor o perfil dos nossos clientes.
- **Características**: capacidades que o projeto deve possuir para atender a uma ou mais necessidades, expressas em linguagem natural e em frases curtas. Exemplo: o projeto deve permitir verificar a média de produtos adquiridos por dia na rede de lojas.
- **Requisitos**: capacidades, qualidades e restrições necessárias para efetivar cada característica. Em BI, o requisito identifica quais informações estarão disponíveis. Exemplo: média de produtos vendidos, tíquete médio, total de vendas, percentual da meta de um produto.

A fase de levantamento apoia-se em definição de templates, reuniões, na técnica **5W2H adaptada** (What, Why, Where, When, Who, How, How Much / How Often), em prototipagem e apresentação ao Key User (mockups) e em validação técnica. O 5W2H é uma ferramenta de gestão baseada em checklist, simples e eficaz como apoio ao planejamento e à resolução de problemas, que ajuda a descobrir o core do projeto e deve ser adaptada ao BI.

Os templates e assuntos importantes citados são: ata de reunião, lista de stakeholders, metodologia de trabalho, visão geral do produto, release plan, definição de requisitos, mapa de fonte de dados, documento de administração do servidor, roteiro para entrevista, desenho do data warehouse e mapa do ETL. Para prototipagem, sugerem-se Balsamiq, Figma e Canva.

3.7 Casos de sucesso

Os slides apresentam casos no formato problema, solução e cenário atual:

- **Gasmig** (Companhia de Gás de Minas Gerais): dificuldade e demora na geração de relatórios; solução por processo de BI para gerenciamento dos dados; hoje, informações com qualidade geradas em tempo real e identificação de oportunidades.
- **Transportadora TNT**: já possuía BI, mas carecia de eficiência e modernidade; apostou no conceito de BI Self-Service com ferramentas SAS Visual Analytics, SAS Visual Statistics e SAS Office Analytics; resultado: menos dependência do setor de TI e modernização das ferramentas.
- **Avon**: decisão baseada em experiência e não em fatos, com informações conflitantes; desenvolveu um data warehouse para análise dos KPIs do negócio; ganhou agilidade, confiabilidade e credibilidade nos dados.

- **UPS Transportes EUA**: dificuldade na gestão de rotas; coleta e estudo de dados de sensores instalados em veículos; economia de combustível, redução de gastos com manutenção e entregas mais rápidas.
- **Fiat**: necessidade de modernizar a Uno; pesquisas em redes sociais e análise dos dados; a nova Uno foi lançada e eleita o carro do ano em 2011.
- **Seleção Alemã**: bons resultados que não se convertiam em títulos; coleta de dados da própria seleção e de adversários por meio de oito câmeras que rastreavam o movimento dos jogadores; melhora de desempenho em velocidade, posse de bola e passes, semifinalista em 2010 e campeã em 2014.

3.8 Estudo de caso: biblioteca universitária

O caso proposto para exercício descreve uma biblioteca central única de uma universidade cujo uso das dependências físicas vem caindo, com alunos de alguns cursos que não a frequentam. A necessidade imediata, com prazo de no máximo um semestre, é compreender o uso atual em comparação com anos anteriores, analisando cursos e dias da semana com maior uso. O sistema transacional possui dados sobre alunos, cursos, livros, autores, editoras e aluguéis, mas seus relatórios não têm flexibilidade nem recursos visuais para análise gerencial. Deseja-se investir em campanhas direcionadas, com dados disponíveis e atualizados semanalmente. Os exercícios pedem identificar o problema atual, a necessidade do negócio, a vantagem do BI sobre o sistema transacional e os dados a observar, além de aplicar o 5W2H e mapear as etapas nas camadas Bronze, Prata e Ouro.

4 Modelagem Multidimensional

4.1 Por que não migrar o modelo transacional

A modelagem multidimensional é a modelagem utilizada para a elaboração do projeto de Data Warehouse, e é completamente diferente da modelagem de dados dos sistemas transacionais. Os dados dos sistemas transacionais não devem apenas ser migrados para o DW porque o modelo transacional é construído obedecendo à **terceira forma normal** e, por isso, não responde com rapidez a questões típicas de consultas de apoio à decisão.

A modelagem multidimensional é mais simples, expressiva e mais fácil de entender, e vai guiar todo o processo de ETL para a construção do DW. Um modelo multidimensional é formado por três elementos básicos: **fatos**, **dimensões** e **medidas**. Ambos os modelos normalmente coexistem em uma organização que tenha um processo de BI.

O comparativo apresentado nos slides:

Aspecto	Transacional	Multidimensional
Foco	Controle do negócio	Gestão do negócio
Conexões	Elevado número de joins	Baixo número de joins
Visão dos dados	Sempre com todos os detalhes	Reestruturados, sumarizados
Modelo de dados	Entendimento mais complicado	Entendimento mais fácil

4.2 Tabela Fato

A tabela fato é uma coleção de itens de dados que correspondem a um item, uma transação ou o evento do negócio que se está analisando. Reflete a evolução do negócio no dia a dia da organização e é um dos pontos mais delicados de um projeto de DW. Trata-se de um assunto sobre o qual é necessário possuir informações históricas para compreensão e tomada de decisão. Exemplos de fato: aluguel, entradas de material em estoque, compras e vendas, pedidos.

Quando existe mais de uma tabela fato no modelo, tem-se uma **constelação de fatos**. Exemplos citados: locadora de veículos e hospedagem em hotel.

4.3 Tabela Dimensão e seus tipos

As dimensões são os elementos que participam do fato, as características daquele fato. São utilizadas como filtros nas consultas e normalmente não apresentam atributos numéricos, pois são descritivas e classificatórias. Exemplos em uma universidade: aluno, professor, sala de aula, disciplina, data de matrícula, campus.

Os tipos especiais de dimensão tratados são:

- **Dimensão degenerada**: não possui relevância suficiente em termos de detalhamento para ser considerada uma dimensão; normalmente é relegada a uma coluna na tabela fato. Exemplo: o código de uma venda.
- **Dimensão conformada**: relaciona-se com várias tabelas fato e normalmente traz todos os campos de que as fatos precisam. Exemplo: dimensão Aluno em uma universidade, ou Cliente em um hotel.
- **Dimensão role-playing**: pode ser usada para vários objetivos. Exemplo: a mesma Dimensão Tempo atendendo várias datas da tabela fato, como data da reserva, data do aluguel e data da devolução de um livro em uma biblioteca.
- **Dimensão junk** (dimensão lixo): inclui atributos que não se encaixam em nenhuma outra dimensão, como flags, indicadores e atributos categóricos. Servem para filtrar ou segmentar dados, mas não para realizar análises. Em vez de criar várias dimensões separadas, agrupam-se em uma única dimensão junk. Exemplos de flags: envio expresso, pagamento online, produto em promoção.

4.4 Hierarquias e métricas

As tabelas de dimensão, em sua maioria, são compostas por **hierarquias de atributos**, que classificam os dados dentro de uma dimensão. Exemplos: Região, Estado, Cidade, Bairro, Loja; Categoria, Subcategoria, Produto; Ano, Mês, Dia. O nível mais detalhado da hierarquia é o **grão** (nível 0), acima do qual se organizam os níveis 1, 2 e assim por diante.

As **métricas ou medidas** são atributos numéricos que representam um fato e estarão sempre na tabela fato. Exemplos: valor em reais das vendas, número de unidades de produtos vendidos, quantidade de estoque, custo de vendas. Os tipos de métrica são:

- **Aditivas**: o valor pode ser relacionado com qualquer dimensão. Exemplo: vendas por data, por produto, por fornecedor, por cliente.

- **Derivadas**: calculadas e geradas a partir de outra métrica; podem ser armazenadas ou calculadas em tempo de execução.
- **Semi-aditivas**: aditivas em todas as dimensões, menos na Dimensão Tempo. Exemplo: saldo em estoque após uma entrada ou saída de produto.
- **Não aditivas**: não podem ser somadas por nenhuma dimensão e são normalmente expressas em percentual. Exemplo: o percentual que uma saída representa sobre uma entrada em uma movimentação bancária.

4.5 Modelo Estrela e Modelo Floco de Neve

O **Modelo Estrela (Star Schema)** tem composição típica com a entidade central fato e um conjunto de dimensões arranjadas ao redor, formando uma estrela. O relacionamento entre fato e dimensão é sempre de um para muitos a partir da tabela fato.

O **Modelo Floco de Neve (Snowflake)** é o resultado da decomposição de uma ou mais dimensões que possuem hierarquias entre seus membros. Ele define relacionamentos muitos para um entre os membros de uma dimensão, formados por relacionamentos entre entidades dimensão de uma hierarquia. É o resultado da aplicação da terceira forma normal sobre as entidades dimensão: um modelo normalizado que evita redundância de valores textuais em uma tabela de dimensão. Um exemplo apresentado decompõe as dimensões Região em Estado e Cidade, e Produto em Tipo de Produto.

4.6 Pontos cardeais e Matriz Dimensão-Indicador

Os **pontos cardeais** são uma metáfora que alude às dimensões presentes no modelo multidimensional e ao seu relacionamento com o fato, ajudando a orientar a definição do modelo. As quatro perguntas são: **Quem? Quando? Onde? O que?**

Um exemplo trabalhado: em uma indústria produz-se uma vasta linha de produtos; a produção de determinado produto ocorre em uma máquina, operada por um funcionário, que produz peças em determinado período de tempo. O fato é Produção; a dimensão de pessoa (Quem) é Funcionário; a de objeto (O que) é Produto; a de lugar (Onde) é Máquina; e a de tempo (Quando) é Data. Outro exemplo mapeia o fato Fornecimento com as dimensões Produto, Data, Filial e Fornecedor.

A **Matriz Dimensão-Indicador** é útil no processo de definição de tabelas fato, principalmente em projetos de médio e grande porte que possivelmente terão mais de uma tabela fato. Suas funções são alinhar e documentar os indicadores-chave do negócio; organizar e visualizar as métricas ou indicadores de desempenho em relação às diferentes dimensões; e fornecer uma maneira estruturada de analisar como os indicadores se comportam em relação a diferentes aspectos do negócio. O documento pode ser construído em uma planilha simples, com as **linhas trazendo os indicadores** (total de vendas, desconto aplicado, custo, lucro) e as **colunas trazendo as dimensões**. Ao observar quais indicadores compartilham o mesmo conjunto de dimensões, identificam-se os agrupamentos que darão origem a cada tabela fato.

4.7 Design patterns e ferramentas

As convenções de nomenclatura adotadas no mercado para os objetos do modelo multidimensional são:

- `SK` — Surrogate Key (chave artificial);
- `NK` — Natural Key (chave natural);
- `DD` — Dimensão degenerada;
- `DT` — Data;
- `NM` — Nome;
- `DIM` — prefixo da tabela dimensão;
- `FT` — prefixo da tabela fato;
- `MD` — prefixo para medida.

Quanto às ferramentas, os slides citam o Astah e o Power Architect, com a diferença de que o Astah é estático enquanto o Power Architect é dinâmico, permitindo gerar os códigos para criar os objetos do projeto. A versão Community do Astah foi descontinuada em 2018.

4.8 Versionamento e a dimensão Data

É possível incluir campos para suportar o **versionamento das dimensões**, que permitem trabalhar com as **SCDs (Slowly Changing Dimensions)**, armazenando histórico e/ou atualizando dados de uma dimensão. A **dimensão tempo é a única dimensão que não pode faltar** em um modelo multidimensional; como boa prática, sugere-se sua geração automática a partir de um script SQL, importando-a em seguida para o modelo criado no Power Architect. O material da disciplina disponibiliza o script `script_dim_tempo.sql` para esse fim.

O exercício proposto pede a construção de um modelo multidimensional para uma loja de calçados que deseja entender melhor o seu estoque, analisando mercadorias, filiais e fornecedores.

5 Tecnologia e Projeto de DW

5.1 Introdução ao Data Warehouse

O Data Warehouse, ou armazém de dados, tem por objetivo disponibilizar informações para a tomada de decisão nas empresas. Um projeto de DW precisa ser diferente de um projeto de banco de dados tradicional: o DW armazena dados atuais e históricos para consultas e análises, a normalização deixa de ser um problema nesse ambiente e nem sempre as informações estarão atualizadas em relação à produção.

Duas definições clássicas estruturam o campo:

- **Bill Inmon**, considerado o pai do Data Warehouse: “Um warehouse (armazém) é uma coleção de dados, orientado a um assunto, integrado, tempo-variante e não volátil, para suporte ao gerenciamento dos processos de tomada de decisão” (*What is a Data Warehouse*, 1995).
- **Ralph Kimball**, que popularizou a abordagem relacional do DW: “Um Data Warehouse consiste em uma cópia de dados de transação, especificamente estruturado para consulta e análise” (*The Data Warehouse Toolkit*, 2002).

Na prática, a construção de um DW consiste na transferência dos dados do ambiente transacional para outro ambiente independente.

5.2 Motivação, características e vantagens

A motivação para o uso do DW vem de dificuldades concretas: dificuldade de recuperação de dados, múltiplos softwares no ambiente operacional, dificuldade de integração entre sistemas, carência de documentação no armazenamento e falta de flexibilidade e agilidade para atender novas demandas.

As características gerais de um DW são: conter apenas os dados necessários para a análise desejada; extrair dados de fontes heterogêneas; transformar e integrar dados antes da carga; requerer máquina e suporte próprios; permitir visualização de dados em vários níveis; extrair (ou não) dados para o Data Mart; ter dados apenas consultados; e ser construído com base no negócio, e não nos sistemas existentes.

As quatro características canônicas de Inmon são detalhadas:

- **Orientado por assunto**: o DW traz apenas os assuntos que interessam para determinada análise. No transacional a orientação é o processo; no multidimensional o foco é a tomada de decisão. Os dados são organizados em torno de tópicos ou áreas de interesse. O exemplo dos slides contrasta um modelo transacional com as entidades Vendas, Clientes e Produtos separadas, e um DW orientado ao assunto Vendas com atributos como `qtdVendida`, produto, data, preço e desconto.
- **Variante no tempo**: os dados pertencem a algum momento específico (snapshot). O DW armazena dados históricos além dos atuais, permitindo rastrear tendências, fazer análises comparativas e identificar mudanças e padrões ao longo de períodos diferentes.
- **Não volátil**: permite apenas cargas (inicial e incremental) e leituras. Os dados são filtrados, limpos e transformados; os dados históricos não são atualizados nem excluídos, o que preserva a integridade do histórico e oferece um registro imutável.
- **Integrado**: refere-se à consistência de nomes e de unidades das variáveis. O exemplo clássico é o atributo sexo, que uma aplicação pode codificar como M/F, outra como 1/0 e uma terceira como H/M; o ETL uniformiza para um padrão único no DW.

As vantagens listadas são: permitir analisar eventos do passado, melhor tempo de resposta, consistência e confiabilidade dos dados, ter um ponto central para relatórios e visualizações, conteúdo de fácil interpretação pelo modelo simplificado e flexibilidade em relação a várias fontes de dados.

5.3 Criação do DW e o exercício com PostgreSQL

A criação do DW normalmente demanda ferramentas de extração e atualização para que o processo seja o mais automatizado possível, apoiando filtragem, limpeza, sumarização e concentração de dados a partir de fontes externas. Os pontos essenciais para o profissional são **SQL** e o conhecimento das bases de dados e do domínio.

O SGBD adotado é o **PostgreSQL**, objeto-relacional de código aberto, administrado pela IDE **pgAdmin**, disponível para Windows, Linux e Mac.

O exercício de exemplo simples parte de um enunciado de negócio: o DW deve trazer informações sobre as vendas, os produtos vendidos e os clientes; não são necessárias informações sobre fabricante

do produto, data de fabricação, quantidade em estoque nem sobre funcionários. A partir de um banco transacional, extraem-se os dados para um esquema à parte denominado DW. Os passos são criar o database `loja`, rodar o script de criação da base transacional, conhecer a base por consultas simples, criar o esquema DW, criar as tabelas produto, cliente e venda, populá-las e testar com consultas.

```
-- Ilustração do fluxo do exercício
CREATE SCHEMA dw;

CREATE TABLE dw.produto (
  sk_produto  serial PRIMARY KEY,
  nk_produto  varchar(20),
  nm_produto  varchar(100)
);

INSERT INTO dw.produto (nk_produto, nm_produto)
SELECT codproduto, nome FROM public.produto;
```

O balanço do exercício: estudou-se o modelo transacional, entendeu-se o problema, criou-se um ambiente à parte para o DW, criaram-se as tabelas, realizou-se a carga e testou-se a base. O que ainda falta, e que será tratado nas aulas seguintes, é realizar o ETL completo, separar a data da venda em uma tabela própria, definir hierarquias, definir granularidade e agregações e usar mecanismos de versionamento. Os exercícios de extensão pedem criar um DW2 incluindo nome e sexo dos vendedores e nome do fabricante de cada produto, e adicionar a tabela `dim_data` relacionando-a com a tabela fato de venda.

5.4 Data Marts e definição da arquitetura

A definição da arquitetura do DW é uma fase importante que guia todo o projeto, e mudanças de arquitetura ao longo do processo são caras. A arquitetura está diretamente relacionada com a empresa e depende de tempo de execução, infraestrutura disponível, ROI, escopo, recursos disponíveis, capacitação dos profissionais, benefícios da utilização dos resultados, velocidade de implementação e amplitude de atendimento.

Data Marts são subconjuntos do Data Warehouse que se concentram em um domínio de negócios específico, como vendas, marketing, recursos humanos ou finanças, projetados para atender às necessidades de análise e geração de relatórios de um grupo de usuários ou departamento específico.

Os **tipos de arquitetura** são:

- **Global**: suporta as necessidades da organização como um todo e é utilizada por todos os departamentos. Traz visões corporativas dos dados, mas exige mais tempo de desenvolvimento e administração e tem custo de implementação alto.
- **Independente**: não há foco corporativo; atende necessidades departamentais sem conectividade com outros. Tem implementação rápida e encanta os usuários, mas não permite visão global.
- **Integrada**: os Data Marts são implementados separadamente, mas há integração. Há aspectos corporativos, os usuários podem acessar dados de outros data marts e há compartilhamento de dados; em contrapartida, é a arquitetura mais complexa.

As **abordagens de construção** são:

- **Top Down**: parte-se do DW corporativo para os Data Marts.

Vantagens	Desvantagens
Herança de arquitetura	Implementação longa
Visão de empreendimento	Alta taxa de risco
Metadados simples	Necessidade de cruzamentos funcionais
Controle de centralização de regras	Aumento das expectativas

- **Bottom Up**: parte-se dos Data Marts para o DW.

Vantagens	Desvantagens
Implementação rápida	Redundâncias e inconsistências
Retorno rápido	Dificuldade em gerenciar os metadados
Foco no problema	Extração crítica pela quantidade de data marts
Redução do risco	Projeto pode virar um "legamarts"

- **Combinada**: modelo único para os data marts, consistência dos dados, data marts evolutivos e coerência entre eles; em contrapartida, data marts inflexíveis, coordenação difícil e tempo maior para a entrega.

5.5 Arquitetura física

A **arquitetura on-premises** oferece controle total sobre tecnologia e governança e atendimento ao compliance, mas as configurações (no-break, backup, customização, implementação, atualizações) e a segurança (incidentes, incêndio, chuva, furtos, tragédias naturais) dependem inteiramente da empresa. Exige alto investimento em hardware e software, principalmente se for apenas para o BI, e a redundância é cara sem garantir segurança total — especialmente se implantada dentro do mesmo ambiente físico. O on-premises traz o conceito de **arquitetura de três camadas**: Top Tier, o front-end para análise dos dados; Middle, a busca de dados em alta velocidade; e Bottom, o servidor de dados.

A **VPS (Virtual Private Server)** tem redundância por padrão, o servidor está fora da empresa, paga-se pelo que se usa (disco, memória, CPU) como um serviço semelhante a água, luz e telefone, não há necessidade de capital inicial, o escalonamento é rápido e toda a manutenção é responsabilidade do provedor.

5.6 Granularidade

A granularidade diz respeito ao **nível de detalhe do DW**. Quanto mais detalhe existir, menor será a granularidade; quanto menos detalhes, maior a granularidade. A granularidade tem impacto significativo no volume de dados do DW — um registro por mês, por exemplo, representa granularidade alta e volume reduzido em comparação com um registro por transação.

6 ETL

6.1 Introdução ao projeto de ETL

ETL (Extraction, Transformation and Loading) é o processo de preparação dos dados para o DW, responsável pelas ações de coleta, limpeza, preparação e carga das interfaces de extração. É importante a validação por profissionais da área de negócio, e o processo precisa ser automatizado, pois a cada nova carga de dados novas transformações precisam ser feitas. O fluxo típico vai das fontes de dados (Data Sources) para a Stage Area, e desta, por meio de múltiplos ETLs, para o DW e os Data Marts.

6.2 Extração

A extração é a retirada dos dados das bases de informações. É uma fase demorada e complexa, já que se pode trabalhar com formatos diferentes de dados para cada interface de extração. Nessa fase deve haver a conversão de dados para um mesmo formato, e o resultado passa pelas transformações necessárias, servindo de entrada para o processamento de transformação. É uma fase fundamental para a clareza e a integração dos dados.

6.3 Transformação

As principais atividades da fase de transformação são:

- Recodificação de categoria, como H/M para M/F;
- Alterações e uniformização de unidades de medida, nomes de campos e datas, como KM/H e MP/H;
- Aplicação de regras ou funções para derivar dados;
- Seleção de colunas;
- Tradução de valores codificados, fase conhecida como limpeza de dados;
- Derivação de valores calculados;
- Junções de dados de diversas fontes;
- Resumo de linhas de dados (sumarização de totais);
- Quebra de um campo em vários;
- Substituição de dados não preenchidos por “Não Informado” ou “N/A”;
- Equiparação de casas decimais;
- Aplicação de regras de localidade;
- Tratamento geral de strings.

6.4 Carga

A carga é a fase em que os dados são transferidos para o Data Warehouse ou Data Mart. Deve ser definida a periodicidade de cada carga; a temporização depende do tipo do negócio, e o momento da carga deve ser aquele de menor utilização dos sistemas que alimentam o DW. O ideal é automatizar o processo de carga via ferramenta.

6.5 Ferramentas de ETL

O material lista as principais ferramentas de mercado: Oracle Data Integrator (ODI), Informatica (Power Center), DataStage (IBM), Talend Open Studio, **Pentaho Data Integration (PDI)**, Microsoft Integration Services (SSIS) e Apache Hop.

6.6 Chave substituta (Surrogate Key)

As chaves substitutas, ou **Surrogate Keys**, são chaves artificiais e auto-incrementais utilizadas para substituir no DW a chave primária original. Podem ser usadas nas tabelas fato e dimensão após a carga do DW; na fato, a chave aponta para a mesma SK da tabela dimensão. Elas resolvem problemas sérios como desempenho e referência a dimensões que se alteram ao longo do tempo (SCDs).

O exemplo dos slides parte de uma tabela Filme transacional cuja chave primária é `codFilme` com valores como ACT-12314, COM-01090 e TER-01617. No DW, essa chave passa a ser chamada de **Natural Key (NK)** e uma nova chave sequencial é criada:

SK_filme	NK_filme	NM_filme
1	ACT-12314	Duro de Matar
2	COM-01090	Austin Powers
3	TER-01617	O Iluminado

6.7 Estudo de caso: integração de duas bases de clientes

Uma empresa deseja carregar sua lista de clientes provenientes de dois sistemas transacionais diferentes para um DW. Os dados já foram carregados em uma stage (processo chamado de **carga da Staging Area**) no database `st_cliente`, com os esquemas:

- `cliente1` (codigoCliente, nome, sobrenome, sexo, datanascimento, cidade, lugar, email, telefone)
- `cliente2` (idCliente, nomecliente, sexo, cidade, email)

O cliente deseja carregar para o DW apenas: código identificador, nome completo, sexo, data de nascimento, cidade e e-mail. A tabela `cliente2` já está no formato esperado e servirá de base para o ETL. As transformações necessárias são:

- **Código identificador**: não poderá mais ser chave, porque o código do cliente se repete ao se tratar de duas fontes de dados — daí a necessidade da Surrogate Key.
- **Nome completo**: a tabela `cliente1` possui os campos nome e sobrenome separados, que precisam ser transformados em apenas um campo.
- **Sexo**: `cliente1` trata o sexo como "Masculino" e "Feminino", enquanto `cliente2` usa "M" e "F"; é preciso integrar.

As estratégias exercitadas nesse estudo de caso são uso de Surrogate Key, transformação de dados, carga de dados no DW e uso de metadados para rastreamento e auditoria.

```sql
-- Ideia central da integração (ilustrativa)
INSERT INTO dw.dim_cliente (nk_cliente, nm_cliente, sexo, dt_nascimento, cidade, email)
SELECT codigoCliente,
       nome || ' ' || sobrenome,
       CASE WHEN sexo = 'Masculino' THEN 'M'
            WHEN sexo = 'Feminino'  THEN 'F' END,
       datanascimento, cidade, email
FROM cliente1
UNION ALL
SELECT idCliente, nomecliente, sexo, NULL, cidade, email
FROM cliente2;
```

6.8 Desnormalização

As dimensões que fazem parte do modelo estrela não podem possuir hierarquias por meio de subdimensões, ou seja, hierarquias explícitas. Quando é necessário utilizar atributos de subdimensões, é preciso **desnormalizar** as entidades que farão parte do DW. O exemplo apresentado desnormaliza uma tabela produto que possui uma hierarquia com categoria: os atributos das duas tabelas passam a compor uma única dimensão, já com os design patterns aplicados.

6.9 Estudo de caso: multas em rodovias e a carga da fato

Um sistema de controle de multas em rodovias serve de base para criar um DW que estuda as ocorrências de multas ao longo do tempo. As tarefas são: criar o modelo multidimensional estrela baseado nos pontos cardeais; criar as tabelas dimensão e fato no DW; definir o que será necessário no processo de ETL; e popular as tabelas do DW. As estratégias exercitadas são desnormalização de dados, carga do DW, prática da modelagem multidimensional, uso de Surrogate Key e cargas manuais.

O ponto pedagógico central é o **entendimento da carga da fato**. A tabela fato é populada a partir do transacional, mas as chaves originais precisam ser traduzidas para as Surrogate Keys das dimensões. Os slides mostram passo a passo os joins que fazem essa tradução:

- `o.placa = dc.nk_carro` — liga a ocorrência à dimensão carro;
- `o.multa = dm.nk_multa` — liga à dimensão multa;
- `o.idlocal = dl.nk_local` — liga à dimensão local;
- `o.data = dt.nk_tempo` — liga à dimensão tempo.

Com todos os joins montados, o SELECT resultante retorna as SKs de cada dimensão e as medidas, prontos para os inserts na tabela fato.

```sql
INSERT INTO ft_ocorrencia (sk_carro, sk_multa, sk_local, sk_tempo, md_valor)
SELECT dc.sk_carro, dm.sk_multa, dl.sk_local, dt.sk_tempo, o.valor
FROM ocorrencia o
JOIN dim_carro  dc ON o.placa   = dc.nk_carro
JOIN dim_multa  dm ON o.multa   = dm.nk_multa
JOIN dim_local  dl ON o.idlocal = dl.nk_local
JOIN dim_tempo  dt ON o.data    = dt.nk_tempo;
```

7 ETL com PDI

7.1 Pentaho Data Integration

O **PDI** é uma ferramenta da suíte Pentaho que facilita e automatiza o processo de ETL. Apesar de fazer parte do Pentaho, pode ser instalado separadamente. É constituído por **Steps**, componentes de transformação que apoiam diversos processos de ETL, conectados por **Hops**, que representam os relacionamentos entre os steps e dão continuidade ao fluxo de dados.

O primeiro exercício prático é criar uma conexão com o banco de dados: cria-se no PostgreSQL o database `teste`, abre-se o PDI, cria-se uma nova transformação e uma nova conexão de database, configura-se e testa-se no botão Test. Para evitar recriar a conexão em cada transformação, o PDI oferece o recurso de **compartilhar a conexão**.

Uma orientação metodológica relevante: embora fosse possível resolver todos os exercícios com uma única transformação, o material opta por quebrar o processo em várias etapas — o que facilita depuração, reuso e entendimento.

7.2 Estudos de caso com o PDI

Os estudos de caso são cumulativos, cada um partindo do resultado do anterior:

- **Estudo de Caso 01**: realizar a extração de dados da planilha `Clientes.xlsx` para uma Staging Area no PostgreSQL.
- **Estudo de Caso 02**: refazer o ETL adicionando tratamento de strings — retirar espaços, capitular os campos NOME e SOBRENOME e colocar o campo EMAIL em minúsculo.
- **Estudo de Caso 03**: a partir da base carregada, criar uma nova base concatenando "nome" e "sobrenome" em um único campo `nomeCompleto`, descartando os campos originais.
- **Estudo de Caso 04**: transformar o campo "sexo", trocando "Masculino" por "M", "Feminino" por "F", "Homem" por "M" e "Mulher" por "F".

Os exercícios que se seguem consolidam a prática:

- **Exercício 01**: criar um novo campo região (Sul, Sudeste, Nordeste, Centro-Oeste e Norte) com base no campo "Estado", usando o step **value mapper**.
- **Exercício 02**: tratar o campo `estcivil` com os tipos D (Divorciado), V (Viúvo), S (Solteiro) e C (Casado), tratando os campos vazios como "Inválido".
- **Exercício 03**: criar uma dimensão local usando o step **Add Sequence**; para a SK, uma sequence iniciando no número 1; para a NK, uma sequence iniciando no número 100. A advertência é explícita: não pode haver linhas duplicadas na tabela dimensão.
- **Desafio**: refazer o exercício 01 aplicando o step **MERGE JOIN**, unindo a tabela CLIENTE4 com a planilha `Regiões.xlsx`.

A progressão desses casos ensina, na prática, os principais grupos de steps: entrada (Excel input, table input), transformação de strings, mapeamento de valores, geração de sequências, junção de fluxos e saída para tabela.

8 Carga Atualização e Versionamento de um Data Warehouse

8.1 O processo de carga das dimensões

Há diversas boas formas de trabalhar com cargas em uma dimensão. O processo adotado na disciplina segue três passos:

1. Acessar as tabelas da stage ou do transacional que formarão a dimensão;
2. Criar um **ETL para cada dimensão**;
3. Inicializar as tabelas com o registro 0, por conta de uma exigência do PDI — são as **Dimensões Não Qualificadas (DNQs)**.

8.2 Estudo de caso completo: o DW da firma

O estudo de caso constrói um DW sobre projetos utilizando os conceitos de SCD, versionamento e atualização. Os passos de preparação são: criar o database `st_firma` e rodar o script correspondente; criar o database `dw_firma`; rodar o script `script_dim_tempo.sql` da dimensão data no `dw_firma`; e rodar o script `criação_dw.sql` no mesmo database.

As **Dimensões Não Qualificadas** podem ser criadas manualmente ou por script; o material opta por script (`dnqs.sql`). A prática recomendada é definir padrões fixos para os valores de preenchimento:

- Textos: `N/A`
- Char(1): `N`
- Números: -1 ou 0
- Datas: 1900-01-01 ou 2199-12-31

Em seguida, cria-se a carga da dimensão funcionário e configura-se o step **dimension_lookup_update**, que é o componente do PDI responsável por buscar o registro na dimensão, decidir se ele é novo ou alterado e aplicar a política de versionamento configurada.

8.3 Tipos de SCD

O material apresenta três comportamentos configuráveis no PDI para cada campo da dimensão:

- **Punch Through**: atualiza o campo SCD, não guarda histórico, não versiona e **atualiza os dados anteriores** — a mudança se propaga para todas as versões existentes.
- **Update**: atualiza o campo SCD, não guarda histórico, não versiona e **não atualiza os dados anteriores** — a mudança vale apenas para a versão mais recente.
- **Insert**: **guarda o histórico**, criando uma nova linha e versionando o dado.

8.4 Os quatro testes de versionamento

A sequência de testes é o coração didático desta aula, pois demonstra empiricamente a diferença entre as políticas:

- **Teste 1 (Update no departamento)**: altera-se o departamento da funcionária de código 1115 na stage com `UPDATE funcionario SET departamento = 'ADM' WHERE codfuncionario = 1115;` e roda-se novamente o ETL. O departamento muda na dimensão, mas **não houve versionamento**.
- **Teste 2 (Insert no departamento)**: configura-se a SCD do departamento para INSERT. Muda-se o departamento para 'TI' e depois novamente para 'ADM', rodando o ETL. Agora **há versionamento**: cada mudança gera uma nova linha na dimensão, preservando o histórico de alocação departamental.
- **Teste 3 (Update no nome)**: com a SCD do nome configurada como UPDATE, altera-se o nome para 'Ana Paula Santos do Nascimento'. O nome é alterado **apenas na última versão** do registro.
- **Teste 4 (Punch Through no nome)**: com a SCD do nome configurada como PUNCH THROUGH, altera-se o nome para 'Ana Paula Santos do Nascimento Soares'. Agora a mudança **se propaga para todas as versões** da funcionária.

A lição prática é que a escolha da política deve seguir a semântica do atributo: atributos cuja evolução é analiticamente relevante (como departamento) pedem Insert; correções cadastrais que devem valer retroativamente (como grafia de nome) pedem Punch Through.

O exercício de fechamento pede realizar a carga das outras duas dimensões, criando um ETL para cada carga.

9 Carga da Tabela Fato e Criação de Pipeline de Dados

9.1 Carga da tabela fato no PDI

A carga da tabela fato é realizada **após TODAS as cargas das dimensões**. O procedimento busca as SKs juntando os dados da stage com os dados das dimensões, usando o step **database lookup/update** para cada dimensão. Para gravar os dados na fato, usa-se o step **table_output**.

O exemplo de carga da fato tem três momentos:

1. Acessam-se os dados da Stage;
2. Buscam-se todas as SKs das dimensões, por meio de um lookup por dimensão;
3. Grava-se o resultado na tabela fato.

Até esse ponto do estudo de caso já haviam sido realizadas as cargas de `dim_funcionario`, `dim_cliente` e `dim_projeto`, cada uma em seu próprio ETL, e a carga de `dim_data` foi feita por script.

9.2 O problema da duplicação

O Estudo de Caso 11 pede calcular o custo da alocação de um funcionário em um projeto, e o Estudo de Caso 12 pede usar a mesma transformação para carregar a tabela fato. Aqui surge um problema clássico e instrutivo: ao reexecutar a carga, **os dados da fato serão duplicados**. A solução adotada nos slides é usar o step **Delete** para apagar os registros duplicados antes da nova inserção, garantindo idempotência da carga.

9.3 Jobs e orquestração

Um **Job** é um recurso do PDI que permite a automação dos processos de carga e atualização do DW. Com o Job é possível criar um fluxo de execução das transformações e também agendar sua execução. O Estudo de Caso 13 pede criar um Job que automatize a carga das dimensões e da tabela fato dos estudos de caso anteriores — na prática, encadeando `ETL01-DIM_FUNCIONÁRIO`, `ETL02-DIM_PROJETO`, `ETL03-DIM_CLIENTE` e, por último, `ETL04-ft_alocacao`.

Como material bônus, a disciplina disponibiliza um vídeo com o passo a passo para realizar uma **carga full** no DW usando o step **combination lookup/update**.

10 Integração de Data Warehouse com Power BI

10.1 Orquestrando a execução do pipeline

Para agendar a execução do Job, basta criar uma tarefa no sistema operacional. É necessário configurar corretamente as pastas do Pentaho e do Job no arquivo `job-automacao.bat`, definir a pasta para o log, definir data, hora e recorrência, e realizar o teste. Esse é o passo que transforma um conjunto de transformações em um **pipeline de dados verdadeiramente automatizado**.

10.2 Conectando o DW ao Power BI

O processo de conexão é simples e realizado da mesma forma como se conecta um banco de dados comum. A diferença é que, após a importação, torna-se possível visualizar a estrela criada no momento da modelagem dos dados, por meio da opção **Visualização do Modelo** do Power BI. O Estudo de Caso 14 pede conectar o `DW_FIRMA` ao Power BI, selecionar as dimensões e a tabela fato e visualizar o modelo estrela. Caso necessário, a porta do PostgreSQL pode ser passada por parâmetro na configuração da conexão.

10.3 Filtragem de dados: eliminando as DNQs

Um ponto de atenção importante: não se pode esquecer de eliminar as **DNQs** criadas para atender à necessidade do Pentaho. Elas existem por exigência técnica da ferramenta de ETL, mas poluiriam as análises se chegassem aos visuais. Para removê-las, utiliza-se a opção de filtros do **Power Query**. O exercício pede realizar o filtro em todas as dimensões, exceto na `dim_data`.

10.4 Identificação das necessidades de um dashboard

O material propõe um roteiro metodológico para a concepção de dashboards:

- **Identificar o público-alvo**: quem será o usuário final — alta administração, equipes de vendas, marketing. A compreensão do público determina quais informações são mais relevantes.
- **Definir objetivos**: qual o propósito do dashboard — monitoramento em tempo real, análise de desempenho. Os objetivos definem quais KPIs e métricas importam.
- **Selecionar métricas e KPIs relevantes**: perguntar o que precisa ser medido para atingir os objetivos e qual o nível de detalhe necessário (dados agregados versus desagregados).
- **Determinar fontes de dados**: banco de dados, planilhas, APIs ou outros sistemas.
- **Escolher elementos visuais**: gráficos de barras, linhas, mapas de calor. A chave é tornar o dashboard intuitivo e fácil de entender.
- **Estruturar o layout**: a disposição deve seguir uma lógica e permitir leitura fácil e rápida das informações mais críticas.
- **Iteração e feedback**: após a primeira versão, coletar feedback e ajustar. Dashboards são frequentemente um trabalho em progresso.
- **Manutenção e atualização**: estabelecer rotina de revisão para garantir que o dashboard continue relevante e útil.

10.5 Checklist para elaboração de dashboards

O checklist consolidado do processo é: levantamento das necessidades do negócio junto aos stakeholders; mapeamento e coleta dos dados necessários; análise do formato e da estrutura dos dados; tratamento e formatação dos dados; definição e criação das métricas analíticas do negócio; definição dos elementos visuais; disseminação das informações aos interessados; configuração do processo de atualização; e definição de políticas de segurança.

10.6 Requisitos do dashboard do estudo de caso

Os doze requisitos definidos para o dashboard do DW da firma são:

1. Visualizar o total gasto com projetos;
2. Visualizar a quantidade total de horas alocadas;
3. Plotar em gráfico o custo de projetos por mês;
4. Total de alocações de funcionários em projetos;
5. Custo médio por alocação;
6. Alocações por dia da semana;
7. Percentual de alocações por cliente;
8. Alocações por projeto;
9. Custo médio de alocação por funcionário;
10. Total de funcionários;
11. Quantidade de alocações por funcionários;
12. Quantidade média de horas por funcionários.

Algumas medidas precisarão ser criadas no Power Query. Uma boa prática é criar uma pasta para organizar as medidas. E há um alerta técnico relevante: é preciso **atenção para contagens nas dimensões com versionamento** — como o SCD do tipo Insert cria múltiplas linhas para o mesmo indivíduo, uma contagem ingênua de linhas superestimaria o total de funcionários.

10.7 Testando o ciclo completo de atualização

O Estudo de Caso 16 fecha o circuito de ponta a ponta: roda-se o script `st_firma_nova_carga` na stage `st_firma`; roda-se o Job no Pentaho; roda-se a atualização no Power BI; e observam-se as modificações nos dados. Esse teste demonstra que o pipeline construído ao longo da disciplina — stage, ETL versionado, carga da fato, job agendado e camada de visualização — funciona de forma integrada e reprodutível.

11 Introdução a NLP

11.1 Do dado estruturado ao não estruturado

Esta aula marca a transição para os **dados não estruturados**. Nos dados estruturados, trabalha-se com tipos numéricos e categóricos, sendo comum representar categorias por códigos numéricos (0 = homem, 1 = mulher, e assim por diante). Mas e quando os dados não têm colunas nem vêm organizados em linhas? As perguntas motivadoras são diretas: como estruturar documentos Word e PDF em colunas de um laudo? Como estruturar e-mails de clientes em colunas? Como estruturar tweets e comentários em bases de dados?

11.2 Text mining

A finalidade do **Text Mining** é processar informações não estruturadas (textuais). As informações podem ser extraídas para obter resumos das palavras contidas nos documentos ou para calcular resumos dos documentos com base nas palavras neles contidas. Assim, é possível analisar palavras e clusters de palavras usadas em documentos, ou analisar documentos e determinar semelhanças entre eles e sua relação com outras variáveis de interesse no projeto de mineração de dados.

O texto não estruturado é muito comum e pode representar a maioria das informações disponíveis para uma organização: e-mails, críticas e opiniões dos clientes, Twitter, laudos, documentos técnicos, livros e resumos.

Uma aplicação destacada é a **análise de respostas de pesquisas** (de mercado ou de políticas). Não é incomum incluir várias questões abertas relativas ao tópico sob investigação, permitindo que os entrevistados expressem suas opiniões sem restrições de dimensões ou formato. Isso fornece informações sobre pontos de vista que não poderiam ser descobertos apenas com questionários estruturados desenhados por especialistas.

O material posiciona o Text mining como **a primeira parte do PLN**, envolvendo o pré-processamento de texto. O Processamento de Linguagem Natural, como ramo da inteligência artificial, usa machine learning para processar e interpretar texto e dados; o reconhecimento de linguagem natural e a geração de linguagem natural são tipos de PLN.

Outras aplicações citadas incluem chatbots automáticos, análise de clientes e textos, respostas em contexto, análise de produtos e **análise de patentes** — com o exemplo de um acervo de 90.000 patentes e de perguntas do tipo "quantos artigos são do autor X?". Um projeto de NLP com grafo de conhecimento na área de petróleo é apresentado, processando texto, imagem e tabela em conjunto, com um exemplo de trecho de laudo técnico sobre sistema de produção e escoamento de um campo de petróleo.

11.3 Pré-processamento de texto

Os filtros e ferramentas do text mining permitem pré-processar o texto para prepará-lo para as aplicações. As etapas são: **tokenização**, **stop words**, **stemmer**, **lematização** e **speech and tag**.

11.3.1 Tokenização

Um **token**, em computação, é um segmento de texto ou símbolo que pode ser manipulado por um analisador sintático, que fornece um significado ao texto; em outras palavras, é um conjunto de caracteres com um significado coletivo. Tokens são os padrões que ocorrem em uma string. Por exemplo, em uma data como 01/09/2017 é possível usar dois tokens para dividir a string em três partes, utilizando a barra como padrão. Assim, qualquer palavra ou frase inserida poderá ser dividida e analisada separadamente. O mesmo pode ser feito com expressões regulares, mas o método de tokens utiliza muito menos processamento e é mais rápido, apesar de não ser tão robusto.

Formalmente, a tokenização secciona um documento textual em unidades mínimas que exprimam a mesma semântica original do texto; essas unidades, os tokens, muitas vezes correspondem a apenas uma palavra do texto.

11.3.2 Stop words

Um dos primeiros passos na preparação dos dados é a identificação do que pode ser desconsiderado nos passos posteriores. É a tentativa de retirar tudo que não constitui conhecimento nos textos, formando uma lista de palavras a serem descartadas — as **Stop Words** ou **Stoplist**. São palavras que não têm conteúdo semântico significante no contexto em que existem, sendo consideradas não relevantes na análise. Normalmente são palavras auxiliares ou conectivas ("e", "para", "a", "eles") que não fornecem informação discriminativa. Geralmente compõem uma stoplist conjunções, preposições, pronomes e artigos. Uma stoplist bem elaborada permite a eliminação de muitos termos irrelevantes.

11.3.3 Word cloud

A **nuvem de palavras** é uma ferramenta visual muito usada para verificar frequência de palavras: quanto maior a frequência, maior a dimensão da palavra na imagem. O material da disciplina inclui um notebook de word cloud aplicado a uma imagem do Cristo Redentor.

11.3.4 Stemming

O processo de **stemming** é realizado considerando cada palavra isoladamente e tentando reduzi-la à sua provável palavra raiz. Isso tem a vantagem de eliminar sufixos, indicando formas verbais e/ou plurais; entretanto, algoritmos de stemming empregam linguística e são dependentes do idioma. Os algoritmos correntes não costumam usar informações do contexto para determinar o sentido correto de cada palavra, e essa abordagem realmente parece não ajudar muito, pois casos em que o contexto melhora o stemming não são frequentes.

O objetivo principal é **reduzir a grande dimensionalidade** das aplicações de mineração de textos: com a remoção de prefixos e sufixos de palavras derivadas de um mesmo radical, que antes seriam consideradas tokens distintos, obtém-se um único token para representar todas elas.

Dois métodos são apresentados:

- **Stemmer S**: método simples no qual apenas uns poucos finais de palavras da língua inglesa são removidos — "ies", "es" e "s", com exceções. Embora não descubra muitas variações, alguns sistemas práticos o usam por ser conservador e raramente surpreender o usuário negativamente.
- **Método de Porter** (Porter, 1980): consiste na identificação das diferentes inflexões referentes à mesma palavra e sua substituição por um mesmo stem. A ideia é agregar a importância de um termo pela identificação de suas possíveis variações, já que termos com um stem comum usualmente têm significados similares — por exemplo, CONSIDERAR, CONSIDERADO, CONSIDERAÇÃO e CONSIDERAÇÕES. É o algoritmo mais comum para stemming em inglês e repetidamente demonstrou ser empiricamente muito eficaz; consiste em cinco fases de reduções de palavras aplicadas sequencialmente, com convenções para selecionar, dentro de cada grupo, a regra que se aplica ao sufixo mais longo.

11.3.5 Lematização

A **lematização** na linguística é o processo de agrupar as formas flexionadas de uma palavra para que possam ser analisadas como um único item, identificado pelo lema da palavra ou forma de dicionário. Na linguística computacional, é o processo algorítmico de determinar o lema de uma palavra com base no significado pretendido. Ao contrário do stemming, a lematização depende da identificação correta da parte pretendida da fala e do significado de uma palavra na frase, bem como do contexto maior em torno dessa frase — frases vizinhas ou até um documento inteiro. Como resultado, o desenvolvimento de algoritmos de lematização eficientes é uma área aberta de pesquisa.

11.3.6 Expressões regulares e POS tagging

As **expressões regulares** são uma ferramenta que elimina expressões comuns em um texto. Podem conter caracteres especiais e comuns; a maioria dos caracteres comuns, como 'A', 'a' ou '0', são as expressões regulares mais simples, pois simplesmente se combinam.

Na linguística de corpus, a **marcação de parte do discurso** (POS tagging), também chamada marcação gramatical, é o processo de marcar uma palavra em um texto (corpus) como correspondendo a uma parte específica da fala, com base em sua definição e seu contexto. Uma forma simplificada disso é comumente ensinada a crianças em idade escolar, na identificação de palavras como substantivos, verbos, adjetivos e advérbios.

11.4 NLTK e análise de sentimentos

O **Natural Language Toolkit (NLTK)** é um conjunto de bibliotecas e programas para processamento simbólico e estatístico da linguagem natural para inglês, escrito em Python. É a biblioteca mais popular de PLN.

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')

from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

texto = "O processamento de linguagem natural analisa textos."
tokens = word_tokenize(texto.lower(), language='portuguese')
sw = set(stopwords.words('portuguese'))
tokens_limpos = [t for t in tokens if t.isalpha() and t not in sw]
print(tokens_limpos)
```

A **análise de sentimentos**, também chamada análise de opiniões ou computação afetiva, tenta classificar textos atribuindo a eles uma orientação, que pode ser **positiva, negativa ou neutra**. A motivação é que cada vez mais as pessoas colocam suas opiniões e sentimentos em serviços disponíveis na web: sites de microblogging como o Twitter, redes sociais e fóruns tornaram-se o meio comum de expressão.

Entre as aplicações de análise sentimental citadas estão a satisfação ou insatisfação com um serviço, a classificação de produtos por satisfação do cliente e campanhas políticas. Outra aplicação mencionada de text mining é o detector de spoiler.

Sobre bibliotecas prontas, o material observa que existem diversos produtos para análise de sentimento, um deles o **TextBlob**, que serve unicamente para inglês com acurácia aceitável; a recomendação é que **sempre é melhor usar um algoritmo de IA**, como será visto nas aulas seguintes.

O material prático da aula inclui notebooks de análise de sentimento em texto e em áudio, análise de sentimento em português, word cloud, bag of words, similaridade TF-IDF e um estudo com reviews de cookies.

O fecho da aula é programático: para poder aplicar algoritmos de inteligência artificial no texto, é preciso **transformar o texto em algo numérico** — e é isso que a aula seguinte aborda, com as representações numéricas de texto.

12 Métodos de apoio a decisão em bases de dados

Material de slides não disponível para esta aula.

O tópico consta da ementa e a aula possui registro em vídeo, mas não há slides no material fornecido. Pelo encadeamento da disciplina, o tema se situa entre a introdução ao NLP e as tarefas específicas de NLP, retomando a ponte entre as bases de dados analíticas construídas na primeira metade do curso e os métodos de suporte à decisão. Os conceitos já trabalhados que sustentam esse tópico são

a distinção entre análise descritiva, preditiva e prescritiva vista em Fundamentos de BI, e os níveis de maturidade em data analytics.

13 Tarefas NLP

13.1 Do texto ao vetor

Esta etapa da disciplina consolida a arquitetura de um projeto de PLN em quatro passos: 1) limpeza dos dados; 2) pré-processamento de dados; 3) **representação**; 4) aplicação de algoritmo de machine learning. Os dois primeiros passos foram tratados na aula introdutória; aqui o foco recai sobre a representação e as operações que ela viabiliza.

O PLN é apresentado como a interseção de vários campos — Ciência da Computação, Inteligência Artificial e Linguística — e é basicamente ensinar computadores a processar a linguagem humana. Tem dois componentes principais: **NLU (Natural Language Understanding)**, o entendimento de linguagem natural, e **NLG (Natural Language Generation)**, a geração. O material observa que a PLN é "AI-completa", pois requer todos os tipos de conhecimento que os seres humanos possuem.

O NLU consiste em derivar significado da linguagem natural: imagina-se um espaço de conceitos (semântico ou de representação) no qual qualquer ideia, palavra ou conceito tem uma representação por computador, geralmente por meio de um **espaço vetorial**. A linguagem natural pode se referir a texto ou fala, e o objetivo de ambos é o mesmo: converter dados brutos em conceitos subjacentes (NLU) e possivelmente na outra forma (NLG).

13.2 Aplicações de PLN

O leque de aplicações apresentado é amplo: machine learning em texto (classificação, regressão, agrupamento), recomendação de documento, identificação de idioma, pesquisa em idioma natural, análise de sentimentos, resumo de texto, extração de significado de palavra ou documento por vetores, extração de relacionamento, modelagem de tópicos, legendagem de imagens, tradução automática, resposta a perguntas e chatbots, além de reconhecimento de entidades (REN), que também pode ser aplicado a textos de redes sociais.

Alguns exemplos concretos:

- **Classificação de documentos**: classificar e-mails como spam ou não spam; classificar críticas de filmes como positivas ou negativas; classificar documentos legais como relevantes ou não relevantes para um tópico.
- **Recomendação de documento**: mostrar as páginas web mais relevantes com base na consulta ao mecanismo de pesquisa; recomendar artigos de notícias com base em artigos anteriores; recomendar restaurantes com base em avaliações.
- **Modelagem de tópicos**: dividir um conjunto de documentos em tópicos no nível da palavra, observando como a prevalência de certos tópicos em uma revista muda com o tempo, ou encontrando documentos pertencentes a um determinado tópico.

13.3 Toolkits

As bibliotecas Python apresentadas são: **NLTK**, a mais popular de PLN; **TextBlob**, que envolve o NLTK e facilita seu uso; **spaCy**, construído em Cython, rápido e poderoso; e **Gensim**, ótimo para modelagem de tópicos e semelhança de documentos.

13.4 Definições: documento e corpus

Duas definições organizam o vocabulário do restante da aula: uma **coleção de palavras** é um **documento** — por exemplo, um tweet; e uma **coleção de documentos** é um **corpus**.

13.5 Representações de palavras

O problema central é que os algoritmos de machine learning recebem números ou categorias nas entradas. Como então colocar palavras em um algoritmo? A resposta é representá-las como números. Assim como os números têm diferentes representações (hexadecimal, decimal, binário), as palavras têm diferentes representações, baseadas em **frequência** e em **contexto**. As duas representações por frequência tratadas são o **Bag of Words / Count Vector** e o **TF-IDF**.

13.5.1 Bag of Words e vetorização de contagem

O modelo **Bag of Words** é uma representação simplificada de texto na qual cada documento é reconhecido como um "saco" de suas palavras. **Gramática e ordem das palavras são desconsideradas, mas a multiplicidade é mantida**. O vetor de contagem nos ajuda a criar uma **matriz de termos de documentos** (matriz termo-documento), na qual as linhas são documentos, as colunas são termos do vocabulário e cada célula guarda a contagem de ocorrências.

```
from sklearn.feature_extraction.text import CountVectorizer

docs = ["o gato dorme", "o gato come peixe", "o cachorro dorme"]
cv = CountVectorizer()
X = cv.fit_transform(docs)
print(cv.get_feature_names_out())
print(X.toarray())
```

13.5.2 TF-IDF

TF-IDF significa *Term Frequency-Inverse Document Frequency* e é uma técnica usada em PLN para avaliar a importância relativa de palavras em um documento. É frequentemente usada em sistemas de recuperação de informação, classificação de texto e mineração de texto.

O TF-IDF é calculado multiplicando a **frequência do termo (TF)** pelo **inverso da frequência do documento (IDF)**. A frequência do termo refere-se ao número de vezes que uma palavra aparece em um documento, enquanto a frequência do documento é o número de documentos em que a palavra aparece. Além da frequência do termo, é preciso considerar quão comum uma palavra é entre todos os documentos: palavras raras devem ganhar peso adicional.

A fórmula apresentada nos slides é:

TF-IDF(termo, documento) = frequência do termo(termo, documento) × log(número total de documentos / número de documentos em que o termo aparece).

A intuição é que o TF-IDF atribui mais peso a palavras raras e menos peso a ocorrências de palavras comuns; informa a frequência com que uma palavra aparece em um documento em relação à sua frequência em todo o corpus; e nos diz que dois documentos são semelhantes quando têm palavras mais raras em comum.

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()
Y = tfidf.fit_transform(docs)
print(Y.toarray().round(3))
```

13.6 Medidas de similaridade

As **medidas de similaridade de texto** são métricas que medem a similaridade ou distância entre duas cadeias de texto. Podem ser feitas pela proximidade da superfície (**semelhança lexical**) das sequências de texto ou pela proximidade de significado (**semelhança semântica**). Medir a semelhança entre documentos é fundamental para a maioria das formas de análise de documentos; entre as aplicações estão recuperação de informações, classificação de texto, agrupamento de documentos, modelagem de tópicos, rastreamento de tópicos e decomposição de matriz.

Duas medidas são estudadas:

- **Semelhança de palavras — distância de Levenshtein**: forma popular de calcular a similaridade de palavras. O TextBlob usa esse conceito para sua função de verificação ortográfica.
- **Semelhança de documentos — similaridade de cosseno**: forma popular de calcular semelhança de documento. O procedimento tem duas etapas: primeiro, colocar cada documento em formato vetorial; segundo, encontrar o cosseno do ângulo entre os documentos. A similaridade do cosseno mede a semelhança entre dois vetores diferentes de zero pelo cosseno do ângulo entre eles.

Para comparar documentos, eles precisam ser colocados na forma de matriz termo-documento, que pode ser feita usando o Count Vectorizer ou o TF-IDF Vectorizer.

13.6.1 Por que ir além do Count Vectorizer

As desvantagens do count vectorizer justificam a adoção do TF-IDF: as contagens podem ser muito simplistas; contagens altas podem dominar, especialmente para palavras de alta frequência ou documentos longos; e cada palavra é tratada da mesma forma, quando alguns termos podem ser mais importantes que outros. Deseja-se uma métrica que explique esses problemas — daí a introdução do TF-IDF.

```
from sklearn.metrics.pairwise import cosine_similarity
sim = cosine_similarity(Y)
print(sim.round(3))
```

Os exercícios propostos acompanham a progressão conceitual: montar a matriz de contagem, converter o mesmo código para TF-IDF, calcular similaridade entre documentos e aplicar a um conjunto de letras de música e a avaliações de hotéis.

13.7 Retomada de classificação

Ao final, o material retoma os **tipos de aprendizado**. No aprendizado supervisionado, os pontos de dados têm resultado conhecido: treina-se o modelo alimentando-o com respostas corretas, e o modelo aprende e finalmente prevê o resultado de novos dados. No aprendizado não supervisionado, os pontos de dados têm resultado desconhecido: os dados são fornecidos ao modelo sem as respostas corretas, e o modelo dá sentido aos dados fornecidos, podendo revelar algo de que não se estava ciente no conjunto de dados.

14 RAG Inicial

14.1 O que é RAG

RAG (Retrieval-Augmented Generation) é uma técnica que combina LLMs com sistemas de recuperação de informação, permitindo que os modelos acessem bases de conhecimento externas atualizadas. Enquanto LLMs tradicionais dependem apenas do conhecimento adquirido durante o treinamento, sistemas RAG consultam documentos, manuais técnicos, normas e dados operacionais em tempo real antes de gerar respostas.

Os benefícios principais são três:

- **Reduz alucinações**: as respostas ficam fundamentadas em documentos verificáveis;
- **Informações atualizadas**: acesso a dados recentes não presentes no treinamento;
- **Customização**: adaptação para domínios específicos.

14.2 Como o RAG funciona

O processo opera em três etapas integradas:

1. **Processamento da consulta**: a pergunta do usuário é analisada e transformada em uma representação vetorial que captura seu significado semântico, permitindo busca inteligente por conteúdo relevante.
2. **Busca de documentos relevantes**: o sistema busca nas bases de conhecimento (manuais técnicos, normas, logs operacionais) os documentos mais relevantes usando **similaridade semântica**, recuperando informações específicas do domínio.
3. **Geração de resposta fundamentada**: o LLM recebe a consulta original junto com os documentos recuperados como contexto, gerando uma resposta que combina compreensão linguística com informações verificáveis e atualizadas.

O exemplo prático dado é uma pergunta sobre a capacidade nominal de um transformador específico: o sistema busca as especificações técnicas daquele equipamento e retorna dados precisos da documentação oficial.

Observe-se a continuidade conceitual com a aula de tarefas de NLP: a "representação vetorial que captura significado semântico" e a "similaridade semântica" da busca são exatamente os conceitos de vetorização e similaridade de cosseno, agora aplicados com embeddings densos em vez de contagens.

14.3 Janela de contexto, chunks e gerenciamento de memória

O material de um projeto aplicado explicita os conceitos operacionais que tornam o RAG necessário:

- **Janela de contexto**: a quantidade de tokens que o modelo pode processar. Se o limite for, por exemplo, 1000 tokens, o texto que ficar fora da janela é "esquecido" pelo modelo.
- **Chunks**: o texto ou documento completo é dividido em blocos.
- **Gerenciamento de memória**: consiste na seleção dos chunks relevantes para preencher a janela de contexto.

Essa é a razão de ser do RAG: como não se pode colocar toda a base documental na janela de contexto, recupera-se apenas o que é relevante para cada pergunta.

14.4 Um pipeline completo de RAG documental

O projeto de automação de fiscalização de projetos apresentado ilustra um pipeline de ponta a ponta, útil como modelo de referência:

Etapa 1 — Transformação dos documentos para Markdown. O fluxo é: documento de entrada, análise das páginas, análise OCR, análise da estrutura do documento, identificação de tabelas, consolidador e, por fim, documento em Markdown.

Etapa 2 — Segmentação do documento. Duas abordagens são comparadas:

- **Segmentação com REGEX**: o documento Markdown passa por um processador que aplica expressões regulares previamente armazenadas, gerando o documento segmentado. Os padrões presentes no documento a segmentar foram identificados e, com isso, as expressões regulares foram criadas.
- **Segmentação com LLM e Prompt Engineering**: o documento Markdown, junto com uma base de dados de prompts, é submetido a um modelo LLM (hospedado em um hub de modelos corporativo), produzindo o documento segmentado. O prompt utilizado especifica o **papel do modelo** e a **tarefa que o modelo deve realizar**.

Etapa 3 — Chunking, vetorização e indexação. Documentos de referência padronizados em texto estruturado (formato TSV por aba) são divididos em chunks; cada chunk é convertido em um **vetor semântico** por um modelo multilíngue, produzindo um **vetor normalizado (cosseno)** com dimensão, por exemplo, de 1024. Os vetores são indexados e armazenados em formato HDF5.

Etapa 4 — Base vetorial. Mantêm-se dois bancos vetoriais: um de referência, com os documentos padronizados, e outro dinâmico, com os uploads do usuário — documentos que passam por segmentação em blocos, embeddings e armazenamento.

Esse pipeline mostra que um sistema RAG real é bem mais do que a chamada ao modelo: envolve OCR, normalização estrutural, estratégias de chunking, escolha do modelo de embeddings, normalização dos vetores para uso do cosseno e uma arquitetura de indexação.

15 LLM Multimodal

15.1 O que são Large Language Models

Large Language Models (LLMs) são modelos de IA treinados em vastas quantidades de texto que compreendem e geram linguagem humana de forma sofisticada. Modelos como GPT-4, Claude e Gemini processam bilhões de parâmetros para entender contexto, raciocinar sobre problemas complexos e gerar respostas técnicas precisas.

Aplicados a um domínio técnico, LLMs funcionam como assistentes capazes de interpretar normas técnicas, analisar documentação, auxiliar em cálculos e gerar código para simulações — democratizando o acesso a conhecimento especializado.

15.2 Multimodalidade: texto, imagem e tabela

A dimensão multimodal aparece de duas formas no material. Primeiro, em projetos de extração de conhecimento que processam simultaneamente **texto, imagem e tabela** para construir grafos de conhecimento. Segundo, no **OCR (Optical Character Recognition)**, tecnologia que converte imagens de texto em dados digitais editáveis e pesquisáveis, permitindo que documentos físicos sejam transformados em formatos processáveis por computador.

O OCR funciona em três etapas:

1. **Pré-processamento**: melhoria da qualidade da imagem por ajustes de contraste, remoção de ruído e correção de distorções, corrigindo problemas como páginas escaneadas em ângulo, manchas e variações de iluminação.
2. **Detecção de texto**: identificação de regiões contendo texto, separando-as de elementos gráficos como linhas, símbolos e imagens; algoritmos segmentam o documento em blocos, identificando parágrafos, títulos e tabelas.
3. **Reconhecimento**: conversão dos caracteres identificados em texto digital usando pattern matching ou redes neurais. Sistemas modernos utilizam deep learning para reconhecer caracteres com alta precisão, mesmo em documentos complexos.

15.3 Análise de estrutura e layout

Algoritmos de OCR modernos vão além do reconhecimento de caracteres, analisando a **estrutura lógica** de documentos técnicos. Os elementos estruturais identificados são títulos e cabeçalhos (por tamanho de fonte, posicionamento e formatação), legendas e notas, blocos de texto e tabelas e listas (preservando relações entre linhas e colunas).

A preservação de hierarquia envolve manter relações espaciais entre texto e elementos gráficos por análise de proximidade e alinhamento; compreender o contexto semântico, sabendo que certas especificações pertencem a determinados equipamentos com base no posicionamento; determinar a ordem de leitura mesmo em layouts complexos com múltiplas colunas; e extrair metadados estruturais de cabeçalho e rodapé. O fluxo de análise estrutural tem quatro passos: segmentação em regiões de interesse, classificação do tipo de cada região, associação de elementos relacionados e estruturação do documento final.

15.4 Desafios e soluções por deep learning

Os principais desafios em documentos técnicos degradados são a **qualidade degradada** (manchas, dobras, desbotamento, baixa resolução), os **símbolos sobrepostos** (linhas cruzando texto, múltiplas camadas de informação) e as **anotações manuscritas** com caligrafia variada, que exigem distinguir texto impresso de manuscrito.

As soluções baseadas em deep learning incluem:

- **Redes neurais convolucionais (CNNs)**: aprendem características visuais robustas que permitem reconhecimento preciso mesmo em documentos degradados, extraindo características hierárquicas que identificam padrões geométricos.
- **Modelos de atenção**: focam em regiões relevantes do documento, ignorando ruído e elementos gráficos irrelevantes.
- **Transfer learning**: aproveitamento de modelos pré-treinados em grandes datasets, com fine-tuning para o domínio específico, reduzindo a necessidade de dados de treinamento.
- **Detectores de objetos (R-CNN e YOLO)**: localizam e classificam múltiplos símbolos simultaneamente com alta velocidade.
- **Vision Transformers**: arquiteturas baseadas em atenção que capturam relações contextuais entre elementos, melhorando o reconhecimento em layouts complexos.

O treinamento apoia-se em datasets especializados com milhares de exemplos anotados, em **data augmentation** (rotação, escala, distorção e adição de ruído artificial para aumentar robustez a variações de qualidade e orientação) e em transfer learning.

15.5 Confiabilidade de LLMs em domínios críticos

Um ponto conceitual importante do material é o tratamento das limitações. Inicialmente, LLMs como o ChatGPT geravam respostas incorretas para tarefas específicas de engenharia elétrica, demonstrando limitações em conhecimento técnico especializado; pesquisadores identificaram que o problema estava na falta de contexto de domínio e de dados de treinamento específicos. A solução implementada teve três frentes: fornecimento de conhecimento de domínio detalhado, uso de casos

práticos e exemplos de problemas reais, e treinamento com dados históricos não vistos anteriormente pelo modelo.

Sobre a confiabilidade, o material registra que sistemas críticos exigem priorização absoluta de segurança e margens de segurança robustas em decisões em tempo real, e que LLMs precisam fornecer soluções confiáveis com **transparência sobre suas incertezas**, permitindo que engenheiros avaliem o nível de confiança em cada recomendação. O framework de confiabilidade proposto tem três componentes: mapeamento de capacidades (forças e fraquezas do modelo em problemas específicos), definição de **níveis de confiança** (quando confiar plenamente e quando requerer validação humana adicional) e transparência de incertezas.

16 Agentes

16.1 O que são agentes de IA

Agentes de IA são sistemas autônomos que percebem seu ambiente através de sensores, processam informações, tomam decisões e executam ações para atingir objetivos específicos, operando com mínima intervenção humana.

Suas características fundamentais são:

- **Autonomia**: capacidade de operar independentemente sem controle humano direto, tomando decisões com base em objetivos e restrições predefinidos.
- **Percepção**: coleta contínua de dados do ambiente por meio de sensores e sistemas de aquisição.
- **Tomada de decisão**: processamento inteligente de informações para determinar ações ótimas considerando múltiplos objetivos e restrições.
- **Ação**: execução de comandos e ajustes no ambiente.

O **ciclo de operação** de um agente tem quatro etapas: 1) perceber, coletando dados do ambiente via sensores; 2) processar, analisando informações e contexto atual; 3) decidir, determinando a ação ótima com base nos objetivos; 4) agir, executando a ação e observando o resultado.

16.2 Tipos de agentes

Os agentes são classificados em quatro tipos principais segundo suas capacidades cognitivas e mecanismos de decisão:

1. **Agentes reativos**: respondem diretamente a estímulos do ambiente sem manter memória de estados anteriores; as decisões baseiam-se apenas na percepção atual. Caracterizam-se por resposta rápida e simplicidade, sem planejamento ou aprendizado. O exemplo dado são relés de proteção que detectam sobrecorrente e acionam disjuntores instantaneamente, sem considerar o histórico.
2. **Agentes baseados em modelos**: mantêm representação interna do mundo por meio de modelos que capturam como o ambiente funciona e evolui. Lidam com observabilidade parcial e raciocínio sobre estados ocultos. O exemplo são estimadores de estado que mantêm um modelo da rede, inferindo valores não medidos a partir das medições disponíveis.

3. **Agentes baseados em objetivos**: planejam sequências de ações para atingir objetivos específicos, avaliando consequências futuras de decisões atuais. Envolvem planejamento, busca e otimização multiobjetivo. O exemplo são sistemas de otimização que planejam para minimizar custos mantendo confiabilidade.
4. **Agentes de aprendizado**: melhoram seu desempenho com a experiência, ajustando comportamento com base em feedback e resultados observados. Caracterizam-se por adaptação, melhoria contínua e aprendizado por reforço. O exemplo são modelos de previsão que refinam a precisão continuamente com novos dados e padrões emergentes.

16.3 Sistemas multiagentes

Múltiplos agentes autônomos podem colaborar para implementar comportamentos que nenhum deles teria isoladamente. O caso trabalhado é o **self-healing** em redes inteligentes, com três fases:

1. **Detecção**: os agentes monitoram continuamente o sistema e identificam falhas por análise de medições e padrões anormais, em milissegundos.
2. **Isolamento**: dispositivos inteligentes atuam automaticamente para isolar a seção com falha, minimizando a área afetada.
3. **Restauração**: o sistema se reconfigura automaticamente por caminhos alternativos, restaurando o serviço para as áreas não afetadas.

A colaboração entre agentes se dá por três mecanismos:

- **Comunicação distribuída**: os agentes trocam informações localmente sobre o estado do sistema, sem depender de controle centralizado.
- **Decisão consensual**: múltiplos agentes negociam e chegam a acordo sobre a melhor estratégia de reconfiguração, considerando restrições de capacidade e prioridades.
- **Execução coordenada**: ações sincronizadas de múltiplos dispositivos garantem transição suave sem causar novas perturbações.

16.4 Aplicações de agentes

O material apresenta diversas famílias de aplicação:

- **Agentes de otimização e despacho**: coletam dados em tempo real, resolvem problemas de otimização multiobjetivo (minimizar custos maximizando confiabilidade e respeitando restrições) e executam ajustes automaticamente, com supervisão humana apenas em situações excepcionais.
- **Agentes adaptativos de proteção e controle**: ajustam dinamicamente parâmetros conforme mudanças na topologia e nas condições operacionais, recalculando a coordenação entre múltiplos dispositivos após mudanças.
- **Agentes de manutenção preditiva**: combinam sensores IoT, análises físico-químicas e inspeção automatizada por drones e robôs com detecção de anomalias por machine learning, priorização de intervenções por criticidade e agendamento otimizado considerando janelas operacionais e disponibilidade de equipes.

- **Agentes de gerenciamento de recursos distribuídos (DERMS)**: plataformas que agregam e controlam recursos distribuídos, tratando-os como uma usina virtual, com coordenação em tempo real de milhares de dispositivos, otimização multiobjetivo e resposta automática a sinais de preço e de rede.
- **Agentes de segurança cibernética**: detecção de anomalias em tempo real em tráfego e comandos, análise comportamental que aprende padrões legítimos para distinguir ameaças reais de falsos positivos, e resposta automática com isolamento de sistemas comprometidos.
- **Agentes de negociação e resposta à demanda**: preveem picos com antecedência, oferecem incentivos personalizados com base em perfil e histórico, e coordenam grandes portfólios de participantes para atingir metas de redução minimizando custos.

16.5 Arquitetura em camadas

Uma arquitetura de referência para sistemas com agentes é apresentada em quatro camadas:

1. **Camada de dispositivos**: os elementos físicos conectados ao sistema.
2. **Camada de comunicação**: protocolos IoT (MQTT, CoAP), edge computing para processamento local e gateways inteligentes garantindo comunicação bidirecional em tempo real.
3. **Camada de IA**: algoritmos de otimização multiobjetivo, modelos de previsão, controle adaptativo e aprendizado por reforço para decisões autônomas.
4. **Camada de aplicação**: interface com o operador, integração com sistemas externos, dashboards de monitoramento e APIs para serviços auxiliares.

As capacidades autônomas resultantes são o **self-healing** (detecção e correção automática de falhas com reconfiguração), a **auto-otimização** (ajuste contínuo de parâmetros para maximizar eficiência e minimizar custos em tempo real) e a **self-configuration** (adaptação automática a novos dispositivos, descoberta de topologia e integração plug-and-play).

Note-se que a camada de aplicação com dashboards de monitoramento fecha o arco da disciplina: o mesmo princípio de disseminação da informação para tomada de decisão que motivou o BI tradicional reaparece, agora sobre uma infraestrutura de agentes autônomos.

17 Síntese da Disciplina

A disciplina percorre um arco coerente que vai do dado estruturado ao dado não estruturado, e do relatório à decisão automatizada.

A **primeira metade** constrói, passo a passo, um projeto completo de Business Intelligence. Começa pela conceituação — a pirâmide DIKW, a definição do BI como processo e não como ferramenta, o ciclo de vida do BI corporativo e as técnicas de levantamento de requisitos com 5W2H, necessidades, características e requisitos. Passa à **modelagem multidimensional**, com fatos, dimensões e medidas, os tipos especiais de dimensão, os modelos estrela e floco de neve, os pontos cardeais e a matriz dimensão-indicador. Aprofunda-se na **tecnologia do Data Warehouse**, com as quatro características de Inmon, os tipos de arquitetura e as abordagens top down, bottom up e combinada, além da granularidade. Detalha o **projeto de ETL**, com suas três fases, a chave substituta e a desnormalização, e o mecanismo de tradução de chaves naturais em surrogate keys na carga da fato. Implementa esse ETL no **PDI**, com steps, hops, tratamento de strings, mapeamento de valores e

junções. Trata a **carga e o versionamento**, com DNQs e as três políticas de SCD — Update, Punch Through e Insert — demonstradas por quatro testes controlados. Automatiza tudo em um **Job** agendado pelo sistema operacional. E entrega a informação em **Power BI**, com filtragem das DNQs, definição de requisitos, criação de medidas e o teste de ponta a ponta do ciclo de atualização.

A **segunda metade** amplia o escopo para os dados não estruturados. Introduz o **text mining** e o pré-processamento — tokenização, stop words, stemming (com os métodos S e de Porter), lematização, expressões regulares e POS tagging — usando NLTK. Passa às **representações numéricas**, o passo indispensável para aplicar machine learning a texto: Bag of Words com vetorização de contagem, TF-IDF, e as medidas de similaridade lexical (Levenshtein) e de documentos (cosseno). Sobre essa base, apresenta as aplicações modernas: **LLMs**, com suas capacidades e suas limitações em domínios técnicos; **RAG**, que resolve o problema da janela de contexto por chunking, embeddings e busca semântica; **OCR com deep learning**, que traz documentos físicos e multimodais para o pipeline; e **agentes de IA**, dos reativos aos de aprendizado, e os sistemas multiagentes com colaboração distribuída.

As conexões entre as duas metades são mais profundas do que a sequência cronológica sugere. A **integração de dados heterogêneos**, aprendida na uniformização de M/F, H/M e 0/1 do ETL, reaparece na normalização de documentos técnicos para Markdown. A **arquitetura em camadas de qualidade** da Medallion Architecture — bronze, prata, ouro — encontra eco no pipeline de RAG, que vai do PDF bruto ao chunk vetorizado e indexado. A **matriz termo-documento** do NLP é, estruturalmente, uma matriz de fatos e dimensões: linhas de observações, colunas de atributos, células com medidas. A **similaridade de cosseno** estudada com contagens de palavras é a mesma operação que sustenta a busca semântica do RAG, agora com embeddings densos. E o requisito de **confiabilidade e rastreabilidade** — que no BI se expressava em metadados para auditoria, DNQs controladas e versionamento por SCD — reaparece nos LLMs como transparência de incertezas, níveis de confiança e fundamentação em documentos verificáveis.

O aprendizado central da disciplina, portanto, não é uma ferramenta nem um algoritmo, mas um método: entender o problema de negócio, modelar os dados de acordo com as perguntas que se quer responder, construir pipelines automatizados e auditáveis que transformem dados brutos em informação confiável, e entregar essa informação de forma que ela efetivamente sustente decisões.