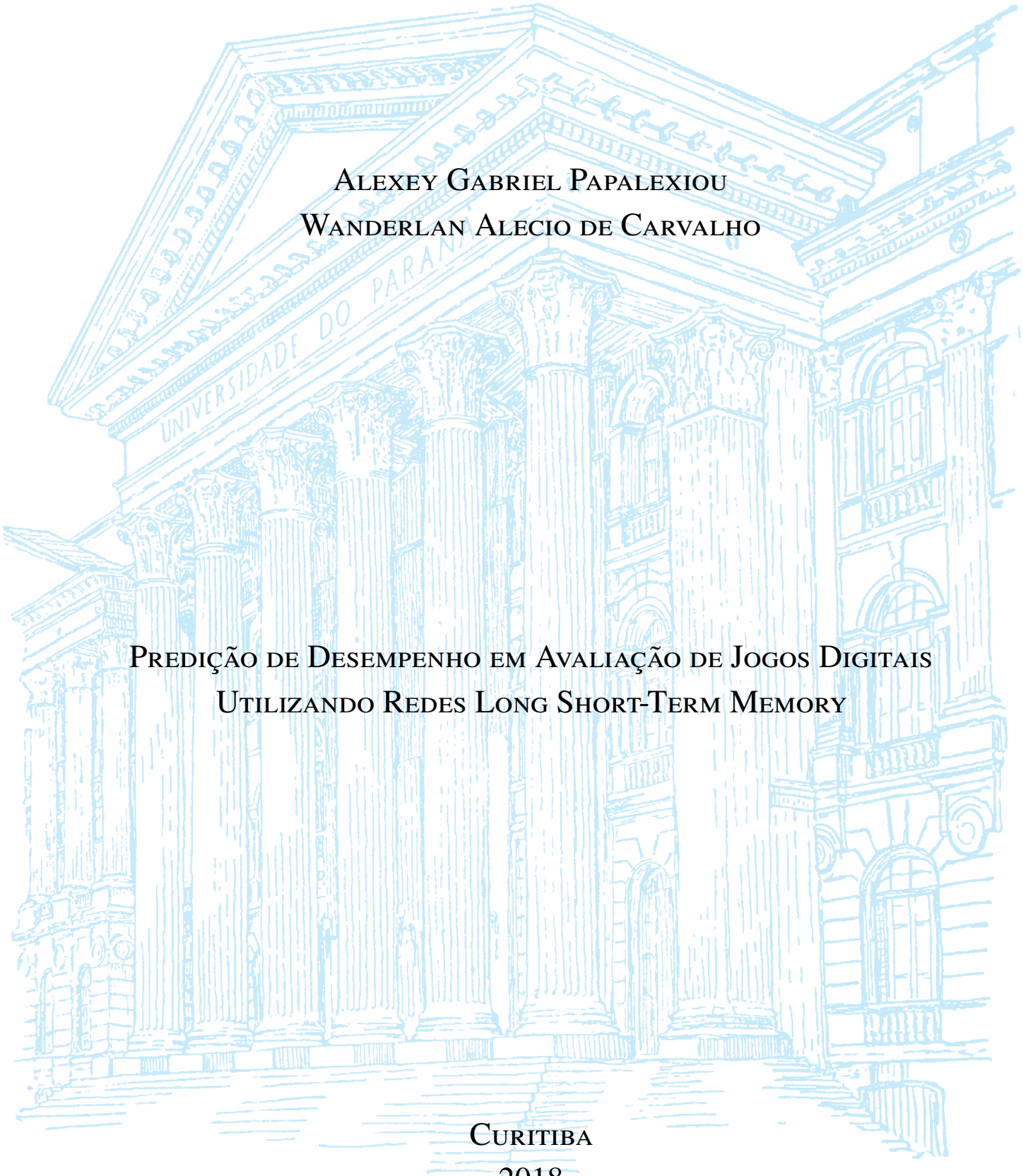
Universidade Federal do Paraná



Alexey Gabriel Papalexiou
Wanderlan Alecio de Carvalho

Predição de Desempenho em Avaliação de Jogos Digitais Utilizando Redes Long Short-Term Memory

Curitiba
2018

Alexey Gabriel Papalexiou
Wanderlan Alecio de Carvalho

Predição de Desempenho em Avaliação de Jogos Digitais Utilizando Redes Long Short-Term Memory

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. Dr. Fabiano Silva.

Curitiba
2018

Ficha catalográfica

Substituir o arquivo 0-iniciais/catalografica.pdf (PDF em formato A4) pela ficha catalográfica fornecida pela Biblioteca da UFPR a pedido da secretaria do PPGInf/ UFPR.

O conteúdo exato da ficha catalográfica é preparado pela Biblioteca Central da UFPR, a pedido da secretaria do PPGINF. Portanto, não "invente" um conteúdo para ela.

ATENÇÃO: por exigência da Biblioteca da UFPR, esta ficha deve ficar no verso da folha de rosto (que contém o nome do orientador e área de concentração). Cuide desse detalhe quando imprimir as cópias finais.

Ficha de aprovação

Substituir o arquivo 0-iniciais/aprovacao.pdf pela ficha de aprovação fornecida pela secretaria do programa, em formato PDF A4.

Resumo

A partir de técnicas de Aprendizado de Máquina, este trabalho possui como objetivo a construção de uma plataforma capaz de predizer o potencial de jogos digitais, ainda em desenvolvimento, de serem bem avaliados por seus futuros compradores quando lançados. Também é discutido o impacto que isso pode ter para ambos os desenvolvedores e investidores do produto. Outros trabalhos com objetivos similares são discutidos, e uma análise sobre fatores capazes de influenciar nas vendas dos jogos é realizada. Um banco de dados de jogos já lançados é criado e experimentos são conduzidos com o objetivo de descobrir o quão bem esta previsão pode ser feita. Foi utilizada uma rede *Long Short-Term Memory*, que recebe como entrada um vocabulário que é representado pelo algoritmo GloVe, e contém ao todo duas sub-camadas, onde a primeira contém 300 células LSTM bidirecionais e uma sub-camada de *dropout*. Os resultados obtidos foram satisfatórios, obtendo uma acurácia média de até 55%, que foi otimizada para 95% após um processo de geração de dados.

Palavras-chave: Jogos digitais, Aprendizado de máquina, Aprendizado profundo, Redes Neurais Recorrentes, LSTM, GloVe.

Abstract

Based on Machine Learning techniques, this work aims to build a platform capable of predicting the potential of digital games, still under development, to be well evaluated by their future buyers when they are launched. It is also discussed the impact this can have for both developers and investors of the product. Other works with similar objectives are discussed, and an analysis on factors capable of influencing the sales of the games is carried out. A database of games already released is created and experiments are conducted with the goal of finding out how well this prediction can be made. We use a Long Short-Term Memory Network, which receives as input a vocabulary that is represented by the GloVe algorithm, and contains in its entirety two sub-layers, where the first one contains 300 bidirectional LSTM cells and a dropout sub-layer. The obtained results were satisfactory, reaching an average accuracy of up to 55%, which was optimized to 95% after a data generation process.

Keywords: Digital games, Machine Learning, Deep Learning, Recurrent Neural Networks, LSTM, GloVe.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

BNF	Backus-Naur Form
GLC	Gramática Livre de Contexto
GPU	Graphics Processing Unit
PLN	Processamento de Linguagem Natural
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
BPTT	Backpropagation Through Time
IPA	Interface do Programa Aplicativo

Sumário

1 Introdução

Este trabalho tem como objetivo o desenvolvimento de um modelo de uma rede neural *Long Short-Term Memory* (LSTM) para classificar o desempenho, em avaliações de usuários, de jogos digitais antes de seu lançamento. Esta ferramenta tem como objetivo ajudar desenvolvedores a direcionar o desenvolvimento de seus jogos e também investidores a poderem identificar produtos com maior potencial de venda.

Este capítulo está dividido em cinco sessões, onde a primeira fala sobre as motivações para este trabalho, a segunda trata dos desafios sobre o problema tratado, a terceira apresenta a solução proposta pelo trabalho, na quarta temos a exposição das contribuições para o estado da arte e na quinta tratamos da organização deste documento.

1.1 Motivação

As décadas de 70 e 80 foram marcadas pelo crescimento dos jogos digitais na cultura popular com o surgimento dos consoles e casas de fliperama. Várias gerações de jogos existiram desde então, sendo possível notar o desenvolvimento da computação como um todo através da evolução desse mercado com a inteligência artificial, capacidade de processamento e conexão em rede sendo as áreas representadas com maior notoriedade nesse meio.

A industria dos jogos digitais é uma das que apresenta o maior crescimento anual. Jogos de diferentes gêneros são lançados diariamente em diferentes plataformas e tipos de mercado. Até o final de 2018, 2,3 bilhões de jogadores no mundo terão gasto 137,9 bilhões de dólares em jogos durante o ano, o que representa um aumento de 13,3% em relação ao ano anterior [New18] e pode ser visto com mais detalhes no infográfico representado na figura 1.1.

Sendo um mercado que gera cada vez mais dinheiro, a capacidade de compreender as técnicas para alcançar o interesse do público e sua eficácia tende a ser uma atividade cada vez mais requisitada mesmo que o mercado para tal ainda seja muito limitado. Porém, estes jogos existem em diferentes estilos, com variações em gráficos, gênero, plataforma, jogabilidade, acessibilidade e classificação etária. Todos esses são elementos que podem influenciar o apelo que o produto tem com o público, e é muito raro que uma única característica sozinha seja motivo o suficiente para justificar o sucesso ou não de um jogo.

1.2 Desafio

Por ser uma indústria voltada completamente para o meio digital, a obtenção de dados se torna uma atividade complexa não pela falta, mas sim pelo excesso de informação. Com a existência de diversas bases de dados, diferentes fóruns de discussões em diferentes línguas, além da veracidade das informações disponíveis na Internet, filtrar informações que sejam realmente úteis para o problema tratado se torna um desafio.

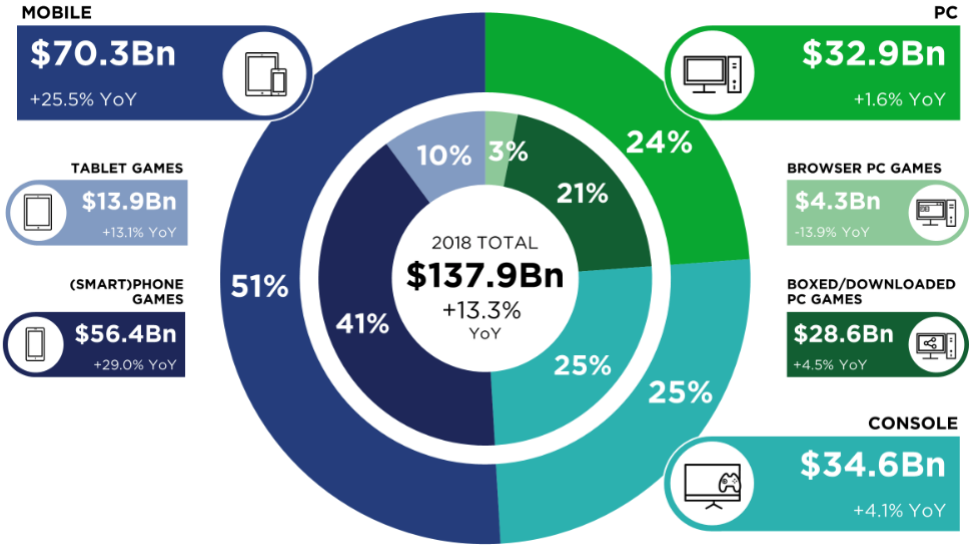


Figura 1.1: Mercado de Jogos Digitais em 2018, com porcentagem de crescimento anual (YoY). Fonte: Newzoo, 2018

Além da dificuldade em filtrar essas informações, outros trabalhos feitos na área encaram o problema sempre dando maior ênfase para variáveis numéricas como número de acessos ou cliques, muitas vezes ignorando características que podem ser sinônimo de interesse ou a falta dele, por parte do público alvo, em determinados títulos.

Outra dificuldade é compreender as motivações por trás de tudo que trás interesse a um jogo. A quantidade de visualizações em uma determinada página não possui significado nenhum se o motivo pelo interesse for negativo e isso traz novamente a ideia de que apenas variáveis que demonstrem a quantidade de pessoas que a publicidade do software atingiu não é sinônimo de algo positivo. Porém, analisar características apenas como avaliações e discussões também não será o suficiente, já que essas podem muitas vezes ter um viés que não demonstra um cenário real.

1.3 Proposta

As técnicas de *Deep Learning* [LBH15] são utilizadas para diferentes tarefas atualmente graças a evolução na capacidade de processamento computacional, que é vista principalmente nas GPUs. A capacidade de aprendizado destas técnicas, junto com a sua flexibilidade para serem usadas em diversos campos de estudo, mostram o potencial desses algoritmos para o futuro da Inteligência Artificial em contraposição a IA clássica, com métodos estatísticos ao invés do uso de métodos heurísticos.

Esse projeto utiliza uma rede neural LSTM [HS97] para prever o desempenho de vendas de jogos com base em variáveis textuais, fazendo uso da *Internet Game Database* [IGD18] como principal fonte de dados para análise e treinamento.

1.4 Contribuição

As principais contribuições deste trabalho são:

- Um modelo de previsão de desempenho em avaliação de jogos digitais fazendo uso de um algoritmo de redes neurais recorrente;

- Um modelo de geração de dados textuais, que visa preservar seu contexto e significado frente a sua representação em vetores de palavras;
- O modelo foi validado por meio de experimentos e apresentamos uma análise de seu desempenho;
- Foram obtidos resultados superiores aos outros trabalhos na área, elevando o *Estado da Arte* em se tratando de técnicas para avaliar o desempenho de jogos quanto a aceitação por seu público consumidor.

1.5 Organização do trabalho

Este trabalho é dividido em seis capítulos onde, no capítulo 2 são apresentados os fundamentos teóricos necessários para a compreensão do trabalho, no capítulo 3 são expostos os trabalhos relacionados que apresentam o estado da arte das pesquisas nesta área em questão, já o capítulo 4 é um aprofundamento na definição do problema. O capítulo 5 explica a metodologia proposta, onde tratamos de falar sobre a base de dados, tratamentos sobre a mesma e o modelo de solução proposta, o capítulo 6 aborda os experimentos e resultados e finalizamos no capítulo 7 com a conclusão do trabalho, fazendo uma análise crítica sobre os resultados obtidos e contribuições para o estado da arte.

2 Fundamentação teórica

Neste capitulo são apresentados os fundamentos teóricos básicos para a composição deste trabalho. Estes fundamentos se dividem nas áreas de PLN [PMN11] e *Deep Learning*, mais especificamente redes neurais recorrentes, que são a base para a implementação deste modelo.

Este capítulo está dividido em duas seções, onde na primeira será feita uma abordagem histórica sobre PLN, destacando a chamada PLN Estatística, suas principais aplicações e como podemos representar dados de séries temporais de forma a processá-los por um computador. Já na segunda seção é feito um estudo sobre Redes Neurais, com ênfase em *Deep Learning*, focado em entender suas principais aplicações e diferenças com o já tradicional processo de *aprendizado de máquina*. Então é explicado o conceito de RNNs (apresentando o conceito de Série Temporal), o conceito do *Vanishing Gradient*, de Redes LSTM e sua aplicação para casos onde as RNNs já não são capazes de processar todas as informações disponíveis, BPTT [Wer90] e suas diferenças com o *Backpropagation* tradicional.

2.1 Processamento de linguagem natural

Com o advento dos computadores, também surgiu a necessidade de se comunicar de forma clara e coesa com essas máquinas, assim surgiu a área de PLN, mais especificamente na década de 1950 como uma intercessão entre as áreas de inteligência artificial e linguística [Tur09].

Após estudos sobre gramáticas de linguagens, foi-se compreendendo o tamanho do problema que era máquinas computacionais interpretarem a linguagem humana [Cho56]. Isso inspirou a criação do *Formalismo de Backus-Naur* [DDM03], que serve como base para representar uma gramática livre de contexto (GLC) e para representar a sintaxe de linguagens de programação.

Apesar das GLCs serem inadequadas para representar linguagem natural, elas são muito usadas na pratica para tarefas de PLN. Como por exemplo na linguagem de programação *Prolog* [BJGJ96], que foi criada com o objetivo de ser aplicada a problemas de PLN.

O tamanho das linguagens, sua natureza ambígua e sem restrições geram dois problemas quando fazemos uma análise por meio de regras simbólicas, como no caso das GLCs: O processamento de linguagem natural deve extrair o significado do texto, e palavras podem ter mais de um significado, por exemplo, o que muitas vezes é impossível de ser representado com regras de restrição; Estas regras não são capazes de interpretar o contexto de um diálogo, como gírias e abreviações.

A partir destes problemas, na década de 1980, Klein [D.05] reformulou os métodos de análise, introduzindo os métodos de aproximação e o uso de técnicas de aprendizado de máquina para uma grande quantidade de dados de linguagem natural. Este foi o início da PLN estatística, que substitui o uso de numerosas regras de restrição, por uma análise de frequência estatística.

Estas abordagens estatísticas geram bons resultados pois, ao aprender com um número grande de dados, estes modelos conseguem generalizar para os casos mais comuns, por isso quanto maior for o número de dados para o treinamento, maior será sua generalização.

Hoje, com o grande interesse na comunicação humano-computador, somado com a viabilização de grandes massas de dados e avanços em hardware e algoritmos, esta tecnologia vem avançando.

O processamento de linguagem natural ajuda o computador a se comunicar com humanos em sua própria linguagem. Por exemplo, torna possível ao computador ler textos, ouvir falas, interpretar, diferenciar sentimentos e determinar partes importantes desses dados.

Hoje as máquinas podem analisar mais dados baseados em linguagem que humanos, sem se cansar e de um modo consistente e enviesado. Considerando a grande quantidade de dados desestruturados gerados diariamente, que vão de registros médicos a mídias sociais, a automação é importante para analisar dados de texto e fala de forma eficiente.

A linguagem humana é espantosamente complexa e diversa, podendo ser expressada de diversas formas, tanto verbalmente quanto por escrita. Existem milhares de linguagens e cada uma tem suas regras de sintaxe e gramática, termos e gírias. Ao escrever, humanos costumam errar ou abreviar palavras, ou omitir pontuações. Quando falam, tem sotaques regionais, resmungamos e são pegos emprestados termos de outras línguas. Por isso as técnicas de PLN ajudam a resolver ambiguidades na linguagem adicionando uma estrutura útil à muitas aplicações, como reconhecimento de voz e análise de texto.

As principais aplicações das técnicas de PLN são:

- **Categorização de conteúdo:** Um resumo de documentos baseados na linguística, incluindo pesquisa e indexação, alertas de conteúdo e detecção de duplicação.
- **Classificação:** Extração de características e captura com precisão de significado para tarefas, como previsão.
- **Modelagem:** Modelagem de texto e fala, de forma que esses dados possam ser compreendidos por uma máquina.
- **Extração contextual:** Retirar informações estruturadas de fontes de dados baseadas em texto.
- **Análise de sentimento:** Capturar o humor ou opiniões subjetivas em grandes quantidades de texto, incluindo o sentimento médio e a mineração de opinião.
- **Conversão de fala para texto e texto para fala:** Transformar comandos de voz em texto escrito, e vice versa.
- **Sumarização:** Geração automática de resumos e sinopses a partir de um texto.
- **Tradução:** Tradução automática de texto ou fala de uma língua para outra.

2.1.1 PLN Estatística

O aprendizado de máquina utiliza algoritmos e métodos estatísticos para possibilitar a separação de padrões sobre dados de exemplo, buscando a generalização para dados fora deste conjunto. O treinamento é feito sobre essa base de exemplos, por meio de métodos iterativos, que visa otimizar um modelo numérico que caracteriza e classifica estes dados, com o objetivo de predizer as classes de novos itens de dados.

Este treinamento pode ser supervisionado, onde já se conhece a classe a que pertence cada item de dado do conjunto de treinamento ou, não-supervisionado, onde o algoritmo de aprendizagem tenta inferir a classificação dos dados automaticamente através de técnicas de agrupamento.

Ambos os métodos de treinamento sofrem com o problema de *overfitting*, ou sobre-ajuste, que é quando um modelo separa muito bem os dados do conjunto de treinamento, porém não consegue classificar dados fora dele, ou seja, o modelo não tem poder de generalização para dados fora do conjunto de treinamento. Este problema ocorre quando, no conjunto de treinamento, as características não distinguem bem os dados, o que pode ser causado por ruido desnecessário nos dados ou por um conjunto muito pequeno de dados para treinamento. Há algumas formas de resolver este problema de sobre-ajuste como o *cross-validation* [K+95], ou validação cruzada, que trata de separar os dados em conjuntos aleatórios e disjuntos de dados de treinamento e validação, onde a precisão final do modelo é estimada por:

$$Ac_f = \frac{1}{v}\sum_{i=1}^{v} \epsilon y_i, \hat{y_i} = \frac{1}{v}\sum_{i=1}^{v}(y_i - \hat{y_i}) \tag{2.1}$$

Onde v é o número de dados de validação e $\epsilon y_i, \hat{y_i}$ é a diferença entre o valor real y_i da saída i e o valor predito $\hat{y_i}$.

Em aprendizado de máquina, há duas principais classes de modelos, os gerativos e os discriminativos e ambos são aplicados a problemas de PLN.

A partir de uma variável observável X e um objetivo Y, um modelo generativo é um modelo estatístico da distribuição de probabilidade conjunta em $X \times Y$, $P(X|Y = y)$. Em relação a classificação, a variável observável X é frequentemente contínua, diferente do objetivo Y que é discreto, pois normalmente consiste de um conjunto de rótulos, e a probabilidade condicional $P(Y|X)$ também pode ser interpretada com uma função não-determinística no formato $f : X \rightarrow Y$. E a partir de um conjunto de dados finitos, um modelo da distribuição condicional $P(X|Y = y)$ é um modelo de distribuição de cada rótulo.

Com os modelos gerativos pode-se gerar dados sintéticos, através de modelos de distribuição de probabilidade, como é o exemplo das *Generative Adversarial Networks* (GAN) [GPAM+14] que pode ser treinada para gerar dados de sequências temporais e os *Autoencoder* [LLS+15] que são utilizados em tarefas de tradução de texto e tarefas de perguntas e respostas, por exemplo.

Diferente do modelo generativo, o discriminativo modela a distribuição $P(Y|X)$, ou o mapeamento direto da variável não observável X para o rótulo Y, dependendo de variáveis observadas nos exemplos de treinamento.

Dado um conjunto de treinamento D, sendo $D = \{(x_i; y_i|i \leq N \in \mathbb{Z})\}$, onde y_i é a saída correspondente à entrada x_i. O modelo discriminativo pode ser divido em duas típicas abordagens. A primeira é o Classificador Linear onde, a partir do comportamento observado no conjunto de dados de treinamento, pode-se simular seu comportamento. Através do vetor de características (x, y), a função de decisão é definida como:

$$f(x, w) = argmax_y w^T \phi(x, y) \tag{2.2}$$

A segunda abordagem seria a Regressão Logística onde, a partir de um problema de decisão binária, o modelo de probabilidade condicional $P(y|x;w)$, onde w é um vetor de otimização do conjunto de dados de treino, pode ser representado por:

$$P(y|x;w) = \frac{1}{Z(x;w)} exp(w^T \phi(x,y)) \tag{2.3}$$

$$Z(x;w) = \sum_y exp(x^T \phi(x,y)) \tag{2.4}$$

A Regressão Logística trata de traçar um comportamento a partir do conjunto de testes, do qual consegue inferir os próximos passos deste comportamento.

Com os modelos discriminativos é possível classificar e separar os dados, a partir de exemplos conhecidos através do treinamento. Alguns exemplos destes modelos aplicados a PLN são as *Suport Vector Machines* [CV95] e as Redes Neurais, que são ambas utilizadas para classificação.

2.1.2 Representação por vetores de palavras

Os dados não estruturados devem ser representados de forma que possam ser entendidos e processados por máquinas computacionais. A não ser que sejam utilizados métodos baseados em árvores, é necessário converter as palavras em conjuntos de vetores numéricos. Esses vetores são a representação de pontos em um tipo de espaço de palavras.

A representação mais simples destes vetores é conhecida com *One-Hot Encoding*[1], que representa cada palavra como sendo um vetor $\mathbb{R}^{|V|\times 1}$ onde a posição que corresponde ao índice da palavra no vocabulário em questão recebe 1 e as outras posições do vetor recebem 0, sendo $|V|$ é o tamanho do vocabulário. Os vetores de palavras neste tipo de representação, seriam como:

$$w^{abelha} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} w^{amora} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} w^{azar} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Nesta representação, cada palavra é uma entidade completamento independente, o que não preserva nenhuma noção de similaridade. Outro ponto negativo desta representação é a alta dimensionalidade, pois para cada palavra do vocabulário existe um vetor de tamanho $|V|$, ou seja, o tamanho da representação do vocabulário seria igual a $|V|^2$, o que não é viável quando falamos de vocabulários com milhões de palavras. Desta forma, uma saída seria buscar a redução deste espaço de $\mathbb{R}^{|V|}$ para algo menor e assim, encontrar um subespaço que codifica a relação entre as palavras.

Outro método seria a Decomposição de Valor Singular (SVD) [KL80], que consiste na acumulação de co-ocorrências de palavras em uma matriz X e então executar uma decomposição de valor singular para obter uma decomposição USV^T. Em seguida são usadas as linhas de U para gerar os vetores para as palavras.

Deixando de lado o processamento de grandes bases de dados, como no caso do SVD, outra abordagem seria criar um modelo que seja capaz de codificar a probabilidade de uma

[1] O termo "one-hot"vem do design de circuitos digitais, significando "um grupo de bits entre os quais as combinações válidas de valores são apenas aquelas com um único bit alto (1) e todos os outros baixos (0) ".

palavra dado o seu contexto. Primeiro, é preciso criar um modelo que atribua uma probabilidade a uma sequência de palavras. Como no exemplo:

"O cachorro pegou a bola"

Um bom modelo dará a esta frase uma probabilidade alta, pois ela está sintaticamente e semanticamente correta, assim como *"Gato o salgado pedra"* deve ter uma probabilidade baixa, pois não faz sentido. Matematicamente é possível representar esta probabilidade em qualquer sequencia de n palavras como:

$P(w_1, w_2, \cdots, w_n)$

Assumindo que a ocorrência das palavras é independente, é possível usar uma abordagem unária para desmembrar esta probabilidade no chamado *Unigram Model* [CPH08]:

$P(w_1, w_2, \cdots, w_n) = \prod_{i=1}^{n} P(w_i)$

No entanto isto é pouco viável, pois a próxima palavra é altamente dependente da sequência anterior de palavras. E o exemplo da frase sem sentido pode ter uma pontuação alta. Portanto uma abordagem mais acurada seria tornar a probabilidade da sequência dependente do par de probabilidades de uma palavra na sequência e a palavra próxima a ela. Esta abordagem é chamada de *Bigram Model* [Tri13]:

$P(w_1, w_2, \cdots, w_n) = \prod_{i=2}^{n} P(w_i|w_{i-1})$

Novamente, este modelo não é ideal pois já que são analisados apenas os pares de palavras vizinhas e não a sentença toda.

Outro tipo de abordagem seria tratar *["O", "cachorro", "a", "bola"]* como contexto e, a partir dessas palavras, ser capaz de prever ou gerar a palavra central *"pegou"*. Este modelo é chamado de *Continuous Bag of Words* (CBOW) [MCCD13]. Este modelo tem como entradas $x^{(c)}$ um conjunto de palavras de contexto em formato *One-Hot* e como saída um vetor *One-Hot* y da palavra central da sentença.

São criadas duas matrizes, $\nu \in \mathbb{R}^{n \times |V|}$ e $\upsilon \in \mathbb{R}^{|V| \times n}$, onde n é o tamanho do vetor de palavras, ν é a matriz de palavras de entrada e υ, a matriz de palavras de saída. Desta forma, este modelo é representado pela figura 2.1.

Outro método para geração dos vetores de palavras seria fazer o inverso do CBOW, ou seja, criar um modelo de tal forma que, dada a palavra central da sentença *"pegou"*, o modelo será capaz de prever ou gerar as palavras circundantes *"O", "cachorro", "a", "bola"*. Nesta abordagem, a palavra *"pegou"* seria o contexto da sentença. Este tipo de modelo é chamado de *Skip-Gram* [MCCD13].

A configuração deste modelo é basicamente a mesma do CBOW, exceto pelo fato de que são trocadas de lugar as variáveis x e y.

2.1.3 GloVe: Global Vectors for Word Representation

Os *Modelos de espaço vetorial semântico da linguagem* [PSM14] representam cada palavra com um vetor de valor real que pode ser usado como recurso em uma variedade de aplicações. As duas principais famílias de modelos para aprender vetores de palavras são:

- Métodos de fatoração de matriz global;
- Métodos de janela de contexto local.

Quando formulado, ambas as famílias sofriam desvantagens significativas. Embora métodos como o *Latent semantic analysis* (LSA) [LFL98] aproveitem de maneira eficiente a informação estatística, eles fazem relativamente mal a tarefa de analogia de palavras, indicando uma estrutura de espaço vetorial sub-ótima. Métodos como *Skip-gram* podem se sair melhor na

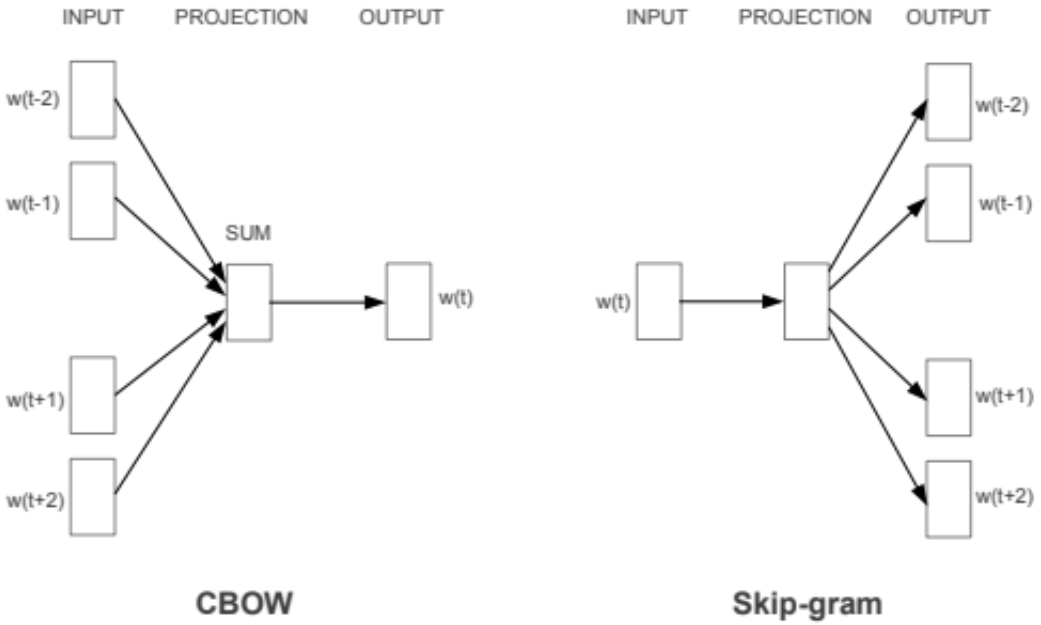


Figura 2.1: Arquitetura do modelo CBOW e Skip-Gram.
Fonte: [MCCD13]

tarefa de analogia, mas utilizam mal as estatísticas do corpus, já que são treinadas em janelas de contexto locais separadas, em vez de em contagens globais de co-ocorrência.

São analisadas as propriedades do modelo necessárias para produzir direções lineares de significado. Também é argumentado que modelos globais de regressão log-bilinear são apropriados para isso. Enfim é proposto um modelo específico de mínimos quadrados ponderados que treina sobre contagens globais de co-ocorrência de palavras e, assim, faz uso eficiente de estatísticas. O modelo produz um espaço vetorial de palavra com desempenho de 75% de precisão no conjunto de dados de analogia da palavra. Também é demonstrado que os métodos apresentados superam outros métodos da época em várias tarefas de similaridade de palavras, e também em um benchmark de reconhecimento.

2.2 Redes Neurais e Deep-Learning

As técnicas de aprendizado de máquina vêm aprimorando importantes ferramentas da computação, desde ferramentas de busca até recomendações de compras em mercados virtuais, além de estarem cada vez mais presentes em aplicações de usuários como câmeras digitais. Sistemas de aprendizado de máquina podem ser utilizados para identificar objetos em imagens, transcrever falas em texto, relacionar produtos a usuários e selecionar resultados de buscas. Dentro dessas aplicações, o uso de técnicas de *deep learning* [LBH15] está em ascensão.

As técnicas convencionas de aprendizado de máquina estavam encontrando limitações na capacidade de processamento de linguagem natural. Por muito tempo o trabalho com essas aplicações envolvia o processo de criação de um extrator de características que fosse capaz de transformar os dados originais da base de dados em uma representação a partir da qual o classificador poderia assim detectar e classificar padrões nos dados inseridos.

O aprendizado profundo utiliza métodos de aprendizagem de representação, que permitem que a máquina receba dados não tratados e automaticamente descubra as representações

necessárias para a classificação. No aprendizado profundo estes métodos possuem múltiplas camadas de representação, obtidas através de módulos não lineares que transformam a representação em cada camada (iniciando com os dados não tratados) em uma nova representação para a próxima camada, como é demonstrado na Figura 2.2. Com um conjunto de transformações suficiente, funções extremamente complexas podem ser aprendidas.

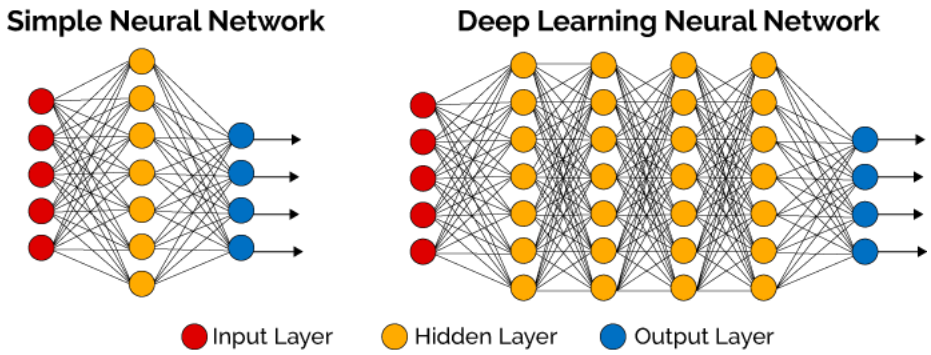


Figura 2.2: Exemplo de arquitetura de uma Rede Neural Deep Learning.
Fonte: https://becominghuman.ai

Estes métodos melhoraram drasticamente o Estado da Arte em várias área de estudo em aprendizado de máquina. Redes convolucionais trouxeram avanços em áreas como processamento de imagem, video e áudio, enquanto as redes recorrentes trouxeram avanços em dados sequenciais como texto e falas.

2.2.1 Backpropagation Through Time

O objetivo do algoritmo de treinamento de Backpropagation é modificar os pesos de uma rede neural a fim de minimizar o erro das saídas da rede em comparação com alguma saída esperada em resposta às entradas correspondentes.

É um algoritmo de aprendizado supervisionado que permite que a rede seja corrigida em relação aos erros específicos cometidos. O algoritmo de treinamento Backpropagation é adequado para o treinamento de redes neurais Feed-Forward em pares de entrada-saída de tamanho fixo.

Já o algoritmo de BPTT [Wer90] é a aplicação do algoritmo de treinamento Backpropagation à rede neural recorrente aplicada a dados sequenciais, como uma série temporal. Conceitualmente, o BPTT trabalha desenrolando todos os passo de tempo de entrada. Cada passo de tempo tem um passo de tempo de entrada, uma cópia da rede e uma saída. Os erros são calculados e acumulados para cada passo de tempo. A rede é recuperada e os pesos são atualizados. Espacialmente, cada passo de tempo da rede neural pode ser visto como uma camada adicional dada a dependência da ordem do problema e o estado interno do passo de tempo anterior é tomado como uma entrada no passo de tempo seguinte.

2.2.2 Vanishing Gradient

O problema do Vanishing Gradient [Hoc98] ocorre quando é preciso treinar um modelo de rede neural usando técnicas de otimização baseadas em gradiente. O que acontece é que à medida em que são adicionadas mais e mais camadas ocultas ao modelo, a velocidade de aprendizado das próximas camadas ocultas continua ficando cada vez mais rápida.

Geralmente, adicionar mais camadas ocultas tende a tornar a rede capaz de aprender funções arbitrárias mais complexas e, assim, fazer um trabalho melhor na previsão de resultados futuros. É aqui que o aprendizado profundo faz uma grande diferença. Devido a quantidade de camadas ocultas que ele possui, é possível compreender e reconhecer dados altamente complexos.

Ao fazer retro propagação, ou seja, se mover para trás na rede e calcular gradientes de perda em relação aos pesos, os gradientes tendem a ficar menores e menores à medida que retrocede-se na rede. Isso significa que os neurônios nas camadas anteriores aprendem muito lentamente em comparação com os neurônios nas camadas posteriores da hierarquia, ou seja, as camadas anteriores da rede são mais lentas para treinar.

2.2.3 Redes Long Short-Term Memory

Dentre as diversas classes existentes de redes neurais, uma RNN possui conexões ponderadas dentro de uma única camada. Oposto a redes Feed-Forward que sempre mandam suas informações para a próxima camada, essas conexões internas da RNN são capazes de armazenar informações através laços, como demonstrado na figura 2.3, assim dando uma memória para a rede. Esta capacidade de armazenamento é útil para um processamento em que dados de uma entrada anterior devem ser considerados. As RNNs são capazes de, com camadas e núcleos o suficiente, implementarem qualquer função computável.

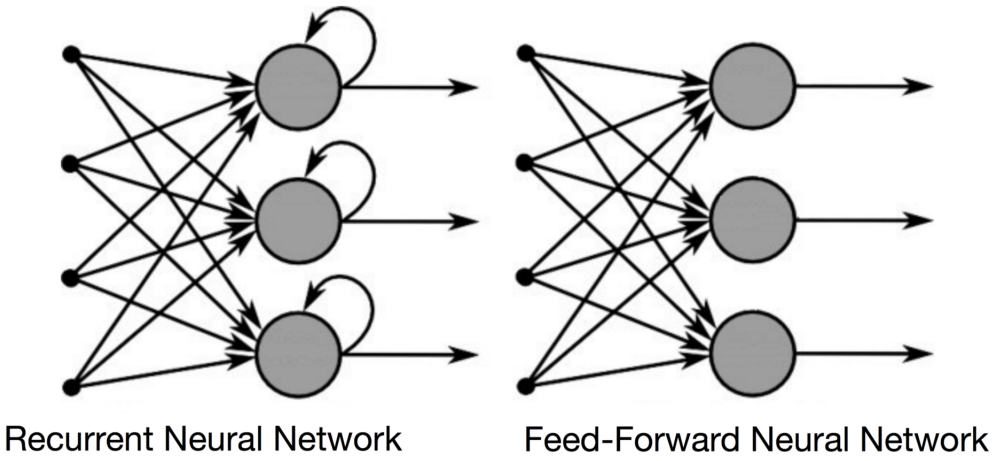


Figura 2.3: Comparação entre a arquitetura de uma Rede Neural Feed-Forward e uma RNN. Fonte: https://towardsdatascience.com

Quando lidando com dados de uma série temporal, um único dado isolado não é útil, já que ele pode não representar completamente se uma série de dados está se alterando. Assim as RNNs se tornam ideais para o processamento de *Sequências Temporais*, já que esta possui capacidade de manter informações a respeito das entradas lidas previamente.

Em alguns casos, é preciso examinar informações recentes para a máquina realizar sua tarefa. Nesses casos, onde a lacuna entre as informações relevantes e o local necessário é pequena, as RNNs podem aprender a usar as informações anteriores. Mas também há casos em que é preciso mais contexto, onde o intervalo entre as informações relevantes e o ponto em que elas precisam se torna muito grande. Infelizmente, à medida que essa lacuna cresce, as RNNs

tornam-se incapazes de aprender a conectar as informações. Esse problema é conhecido como Short-Term Memory.

Dentre as topologias de RNNs, a *LSTM* [HS97], exemplificada na figura 2.4, vem sendo usada com redes convolutivas para descrever o conteúdo de imagens e vídeos. Essa topologia de rede tem como comportamento padrão aprender dependências de longo termo, resolvendo o problema do Short-Term Memory. Isso é obtido por um algoritmo, baseado em gradiente, para uma arquitetura que imponha fluxo de erros constantes através de estados internos de unidades especiais. Sua característica principal é ser capaz de "lembrar"o que realmente faz diferença para a classificação da sequência e "esquecer"informações irrelevantes. Este comportamento se da por meio de seus *gates* e pelo uso da função *sigmoid*, que normaliza os dados entre 0 e 1, e serve como um controlador de fluxo para a sequencia de entrada.

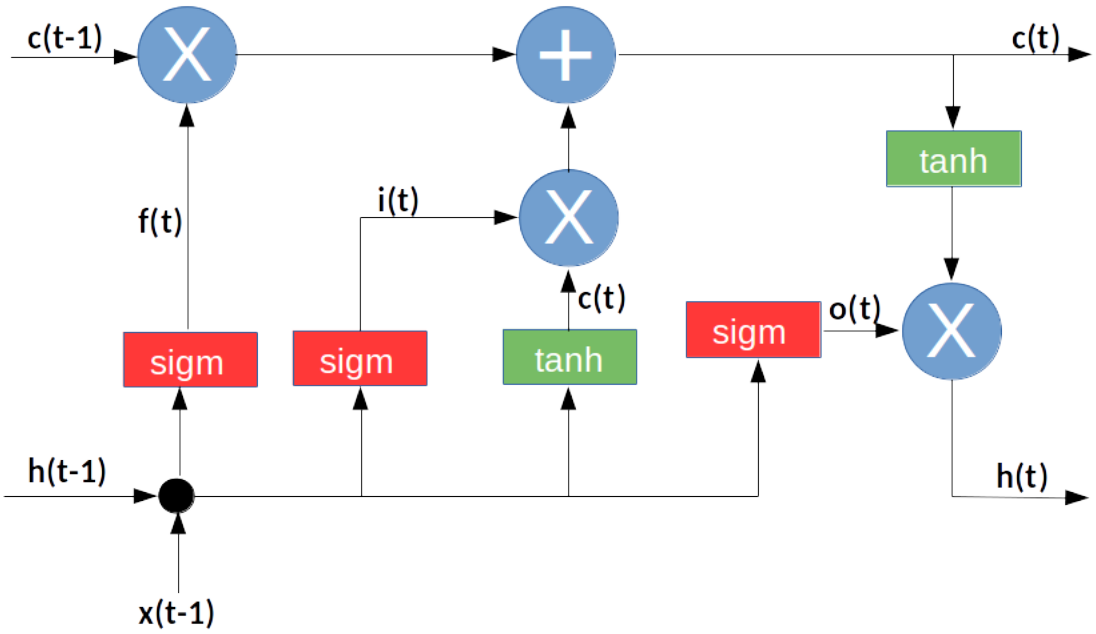


Figura 2.4: Representação da célula LSTM.

Ao todo, a LSTM é representada por três *gates* e dois estados:

- **Forget gate:** Este portal, $f(t)$ decide quais informações devem ser descartadas ou mantidas. As informações do estado oculto anterior $h(t-1)$ e as informações da entrada atual $x(t-1)$ passam pela função sigmoide. Os valores surgem entre 0 e 1. Quanto mais próximo de 0 significa esquecer, mais próximo de 1 significa manter.

- **Input gate:** Para atualizar o estado da célula $c(t)$, tem-se o portal de entrada $i(t)$. Primeiro, o estado oculto anterior $h(t-1)$ e a entrada atual $x(t-1)$ passam pela função sigmoide, que decide quais valores serão atualizados, transformando os valores entre 0 e 1. 0 significa não importante e 1 significa importante. Pode-se também passa o estado oculto $h(t-1)$ e a entrada atual $x(t-1)$ para a função de tangente hiperbólica para normalizar valores entre -1 e 1 para ajudar a regular a rede. Então saída da tangente hiperbólica $c(t)$ é multiplicada com a saída da sigmoide. A saída sigmoide decidirá qual informação é importante para manter na saída do tangente hiperbólica.

- **Cell state:** Com informações suficientes para calcular o estado da célula $c(t)$, primeiro, o estado da célula $c(t-1)$ é multiplicado pela saída do *forget gate* $f(t)$. Isso tem

a possibilidade de descartar valores no estado da célula se ela for multiplicada por valores próximos a 0. Então usa-se a saída do *input gate* $i(t)$ para fazer a adição $c(t) = c(t-1) * f(t) + i(t) * c(t)$, que atualiza o estado da célula para novos valores que a rede neural acha relevantes.

- **Output gate:** Por último, há o portal de saída $o(t)$. O portal de saída decide qual deve ser o próximo estado oculto $h(t)$. O estado oculto também é usado para previsões. Primeiro, o estado oculto anterior $h(t-1)$ e a entrada atual $x(t-1)$ passam em uma função sigmoide. Então o estado da célula recém-modificado $c(t)$ passa pela função de tangente hiperbólica. A saída da função de tangente hiperbólica é multiplicada com a saída da sigmoide $h(t) = c(t) * o(t)$, para decidir quais informações o estado oculto deve carregar. A saída é o estado oculto. O novo estado da célula e o novo oculto são então transportados para a próxima etapa de tempo.

3 Trabalhos Relacionados

Este capítulo apresenta trabalhos relacionados, que representam o estado da arte em se tratando de métodos para predizer o sucesso, vendas e/ou notas de jogos digitais.

Este capítulo é dividido em três seções. Na seção 3.1 temos um trabalho que tenta prever o sucesso de vendas de jogos a partir de fóruns de discussão online. Já em 3.2 é apresentada uma dissertação de mestrado com foco em tentar dizer se um jogo ainda não lançado fará sucesso com base em informações obtidas na Internet, a maior plataforma de jogos digitais para computadores pessoais. Para concluir é feita uma breve discussão sobre os dois trabalhos, apresentado uma análise crítica.

3.1 Predicting Video Game Sales Using An Analysis Of Internet Message Board Discussions

Como a quantidade de texto publicada por usuários vem crescendo, se tornou humanamente impossível de acompanhar todas as discussões sobre um determinado tema na Internet. O estudo *Predicting Video Game Sales Using An Analysis Of Internet Message Board Discussions* [Ehr11] tenta determinar o que comunidades de jogos digitais estão discutindo utilizando métodos estatísticos para criar um modelo de dados obtido desde discussões até dados de venda, tentando prever quais jogos a comunidade comprará.

Tomando como ponto de partida que jogos que estão sendo mais discutidos são os que mais vendem, assim estabelecendo uma relação direta entre o quão discutido o jogo é com a quantidade final de vendas do mesmo para tentar criar um modelo que possa ser usado comercialmente para predição de vendas. Entre as principais funcionalidades deste modelo listadas no estudo estão:

- O departamento de marketing de uma empresa para rastrear os efeitos de uma estratégia de marketing de uma rival.
- Estudar com quais grupos sociais o jogo é mais popular para fazer um bom uso do orçamento de divulgação.
- Se um jogo fizer mais sucesso do que o esperado, para determinar se devem ser produzidas mais cópias do produto para suprir a demanda.

As limitações deste estudo são, principalmente, a disponibilidade limitada de dados tanto de venda como de discussões. Em se tratando de dados de venda, o estudo alega que as duas principais fontes de venda de jogos digitais são o NPD Group e o VGChartz, a primeira sendo paga e a segunda gratuita. Já quanto a dados de discussões textual a análise de dados é considerada complicada por conta da ausência de fontes a respeito de quem está discutindo, dificultando a classificação dos comentários.

Na revisão da literatura, o autor menciona estudos envolvendo predição de sucesso de filmes através do IMDB, predição do futuro através de redes sociais, estudos explorando o valor de avaliações e discussões online além de artigos sobre previsões sobre o mercado de ações através de informações disponíveis, também, em redes sociais e discussões online.

O estudo é divido em 3 partes:

- Coletar os dados de uma comunidade online.
- Formatar e analisar os dados coletados
- Criar um modelo para prever o desempenho de jogos individuais.

Para a primeira parte foi escrito um software capaz de automaticamente baixar e compilar os dados. Na segunda parte, foi escrito um programa que usa métodos de processamento de linguagem natural para ler e analisar os dados coletados. A saída desse programa é uma lista dos jogos que mais estão sendo discutidos na comunidade investigada, junto com um número para classificar o quanto cada produto está sendo discutido. Na terceira e última parte é utilizada a análise obtida no passo anterior, junto com dados sobre vendas e lançamentos de jogos obtidos na internet para criar um modelo que prediz o quão bem cada jogo digital provavelmente será vendido.

O tratamento de dados consistiu em dividir as informações de discussões em grupos, com cada grupo representando uma semana dentro de um período de tempo. Devido a grande quantidade de dados, apenas 10% dos dados de cada semana foram utilizados. Toda essa coleta de dados é feita através de um programa escrito em Python que consegue separar o texto dos usuário de todo o resto das informações obtidas em cada página do NeoGAF, fonte de onde todas as informações de discussões foram retiradas.

Na análise dos dados outro software escrito em python analisa os dados para saber quais jogos estão sendo discutidos e o quanto cada um é mencionado. O maior problema encontrado nesse estágio é que como os jogos ainda não foram lançados, é difícil de prever o nome desses jogos. Para tal, o autor usa uma lista de lançamentos da semana e uma lista de jogos que apareceram na lista gerada para a semana anterior e combina as duas na lista para a semana atual.

Outro problema encontrado foi que usuários diferentes referenciam o mesmo jogo de modos diferentes, além de erros humanos de digitação. Esse problema é resolvido através do uso de expressões regulares, porém o resultado não é perfeito.

As informações obtidas são então formatadas em uma estrutura de dados para ser usada em um software de aprendizado de máquina que irá usar esses dados para criar um modelo capaz de prever como será a venda de um jogo digital.

3.2 Machine Learning for Predicting Success of Video Games

O estudo *Machine Learning for Predicting Success of Video Games* [Trn17] tenta prever o sucesso de jogos com base em dados disponíveis, normalmente, antes do lançamento do produto

O mercado de jogos digitais, que vem apresentando grande crescimento desde seu surgimento no início dos anos 1970, alcançou a margem de lucro de 16.5 bilhões de dólares em vendas nos Estados Unidos da América no ano de 2015. O mercado para jogos de computador viu o aumento de suas vendas digitais na plataforma da empresa Valve, a Steam Store graças a um software chamado Greenlight que permite que desenvolvedores lancem versões digitais de seus jogos na loja digital.

Com o aumento do mercado, que já contava com mais de 10 mil títulos apenas nessa plataforma em 2016, se torna cada vez mais difícil para desenvolvedores se destacarem e venderem cópias o suficiente de seus jogos para financiar o desenvolvimento deles. Com a possibilidade de se prever o sucesso que um jogo fará, o desenvolvimento do mesmo poderia ser adaptado para atrair uma quantidade maior de compradores ainda antes do lançamento, ou até mesmo cancelar o desenvolvimento de um título para evitar desperdício de dinheiro.

Este estudo é pioneiro em tentar calcular o sucesso de jogos antes do seu lançamento, utilizando dados recentes que tenham mais conexão com a realidade do mercado. O objetivo do trabalho é criar uma base de dados de jogos para computador pessoal e estimar o sucesso de um jogo baseado apenas em informações descritivas, normalmente já conhecidas antes da data de lançamento do produto final.

Na seção de trabalhos relacionados são mencionados diferentes estudos envolvendo aprendizado de máquina com jogos digitais e também 3 estudos sobre previsão sobre vendas de jogos.

A proposta feita no trabalho pode ser dividida em 4 principais partes:

- Fatores conhecidos que afetam o sucesso de um jogo;
- Coleta de dados;
- Preparação dos dados coletados;
- Experimentos conduzidos e seus resultados.

Ao analisar os fatores já conhecidos por afetarem o sucesso de jogos, é mencionado que avaliações de revistas de jogos perderam sua importância por conta da ascenção de sistemas de vídeo em demanda e serviços de transmissão em tempo real como Youtube e Twitch. O número de potenciais compradores tomam decisões a partir desses novos tipos de conteúdo. Entre os principais fatores, são classificados no estudo como mais importantes:

- Cliques em ferramentas de busca;
- Análises feitas em plataformas online;
- Serviços de vídeo;
- Discussões em comunidades online.

Já na coleta de dados, é mencionado que na época em que o estudo foi feito não existia nenhuma ferramenta para ler a base de dados de jogos lançados da plataforma Steam. Devido a isso, o autor criou sua própria base de dados, contendo informações de pré-lançamento de jogos da Steam. A maioria dos dados obtidos são informações que não mudam após o lançamento, ou foi possível, para cada jogo, adquirir a informação equivalente a época em quem o mesmo foi lançado. Como a Steam não fornece informações a respeito das vendas feitas na plataforma, foram utilizados dois serviços alternativos, Steam Charts e Steam Spy. Como jogos podem ser lançados por equipes de desenvolvedores de diversos tamanhos, se torna muito difícil decidir se um jogo fez sucesso ou não, portanto o que é analisado é a quantidade de vendas e a média de jogadores concorrentes após dois meses do lançamento.

Após a coleta dos dados, é preciso prepará-los limpando dados inúteis e reunindo tudo em dois conjuntos de dados, um contendo todos os jogos obtidos na coleta e outro com todos os dados necessários para fazer as análises. Todos os campos obtidos para formar a base de dados tem relação com os tópicos abaixo:

- Jogabilidade;
- Produtora do jogo;
- Interação com funcionalidades da Steam;
- Requisitos do sistema;
- O título ser sequência de um jogo anterior;
- Linguas disponíveis;
- Plataforma Windows.

Para conduzir os experimentos, os dados são divididos em grupos para treinamento (60%), validação (20%) e teste (20%), com os dados sobre jogos mais antigos no grupo para treinamento e mais recentes para teste. Com a grande variação na quantidade de jogos concorrentes é implementado um método de regressão para ser possível apontar se um jogo fez sucesso ou não. Os jogos são divididos em grupos que alcançaram até 10, 100 ou que ficaram acima 100 jogadores simultâneos. Após serem feitas comparações entre os modelos de aprendizado SVM e Random Forest, o resultado de nenhum dos métodos foi considerado satisfatório. Após os teste foi notado que jogos feitos por produtoras que já possuíam pelo menos dois outros lançamentos anteriores apresentaram resultados mais satisfatórios, além de uma forte relação entre as principais características conhecidas pré-lançamento com o número de jogadores 2 meses após o lançamento.

3.3 Discussão

Ao analisar o desempenho dos dois trabalhos resumidos anteriormente nesta seção, foi fácil notar que a maior falha cometida nos dois, não está nos experimentos conduzidos, mas sim nos dados. As duas propostas tiveram resultados não ótimos por cometerem erros nessa fase do processo que, mesmo sendo a fase inicial, afeta diretamente o resultado final. Não é possível um bom aprendizado com uma base de dados que não represente o cenário estudado de forma coerente.

No trabalho resumido na seção 3.1, o próprio autor ressalta que foram analisados apenas quadros de mensagens de um fórum, e que esse fórum não representa, necessariamente, a opinião do público alvo do produto. Porém mesmo dentro desse fórum, a quantidade de mensagens que são obtidas, em relação ao total, é baixa. Em nenhum trecho do texto é mencionado qualquer tipo de análise que argumente a probabilidade desses dados representarem a realidade do próprio quadro de mensagens de onde eles são obtidos. Para fazer essa análise obtendo todos os comentários de um fórum online já exigiria uma capacidade de processamento que, por enquanto, não é possível. Se essa solução for trazida para um ambiente em que você tenta analisar a reação de toda a comunidade alvo do produto, a capacidade de processamento necessária se torna ainda maior.

Ao se analisar o trabalho da seção 3.2 também se observa um problema na coleta dos dados, já que todos eles são obtidos de uma plataforma que não oferece todas as variáveis interessantes para tratar o problema, e ainda limita a obtenção desses dados para o usuário. Ainda assim, este problema acaba não sendo tão grave já que é compensado pela cofiabilidade da fonte, o que se prova ao fazer uma comparação de resultados com o trabalho da seção 3.1. No

qual o maior problema se encontra na forma em que a informação obtida é tratada antes de os experimentos serem conduzidos.

Após o autor fazer uma breve análise sobre os elementos que mais influenciam o desempenho do sucesso de um jogo, o modo como os dados são tratados não demonstra levar toda essa análise em consideração. Ao mesmo tempo que o autor dá grande ênfase a ferramentas de busca, serviços de vídeo e comunidades online, ele mantém seus dados analisados com foco em características do jogo, e não no engajamento online que o mesmo cita como maior influência. Além disso, usar dados de jogos mais antigos para treinamento e recentes para teste é uma abordagem que não foi coerentemente justificada. Jogos do gênero *MMORPG*[1] apresentaram mudanças muito menores no modo como os jogos são divulgados enquanto no gênero *FPS*[2] os métodos de publicidade se tornaram muito mais intensos. Portanto, testar dados novos em um treinamento feito com informações mais antigas, não é garantia nenhuma de que o modelo está bem treinado para o caso de teste. A simples aleatorização dos dados antes de iniciar o treinamento, incluindo a data de lançamento de cada jogo como variável, é uma abordagem que deixaria a amostra de dados melhor dividida.

Nos dois principais trabalhos analisados, os maiores problemas detectados foram na preparação e, principalmente, coleta dos dados. Com o aumento constante da capacidade de processamento das GPUs e os grandes avanços realizados na área de Aprendizado de Máquina nos últimos anos, fica bem claro que atualmente o maior desafio para muitos cientistas na área está relacionado aos dados que serão utilizados. Ao não possuir os dados corretos para o problema a ser tratado, os trabalhos não alcançarão um resultado satisfatório.

[1] Massively Multiplayer Online Role-Playing Games (MMORPGs) são uma combinação de jogos de "interpretação de papéis"e jogos online multi-jogador em que um grande número de jogadores interagem entre si dentro de um mundo virtual.

[2] Tiro em Primeira Pessoa, do inglês First-Person Shooter (FPS), é um gênero de videogame centrado em torno de armas e outros combates baseados em armas em uma perspectiva em primeira pessoa.

4 Definição do Problema

Apenas uma fração dos jogos lançados a cada ano conseguem fazer sucesso. Porém, nem todos esses jogos são mal avaliados, alguns simplesmente não conseguem a publicidade necessária para se tornarem um produto gerador de lucro, e mesmo sendo produtos de qualidade acabam fracassando nas vendas. Um jogo com potencial para atrair avaliações positivas e, consequentemente, interesse do público, acaba se tornando uma boa oportunidade de investimento. Com a constante ascensão do mercado dos *games* é de se esperar que um produto capaz de atrair a curiosidade de várias pessoas e agradar aqueles que o adquiram possa trazer lucro para investidores.

Levando isso em conta, a habilidade de prever a receptividade de um jogo é interessante tanto para os desenvolvedores, que poderão usar isso para vender seu produto, quanto para investidores que poderão fazer uso de tal ferramenta para melhor escolher onde investir seu dinheiro.

Neste capítulo são explicados os aspectos que ajudam a definir qual o problema e a sua importância, para esses dois grupos, de uma ferramenta capaz de prever o desempenho, em avaliação, de jogos digitais.

4.1 Investimento e Financiamento

Um dos maiores problemas que toda nova empresa de desenvolvimento de jogos enfrenta é a falta de financiamento. Com a ausência de uma ferramenta de busca eficiente para a análise de investimento e financiamento dessas empresas, acumulada com a quantidade cada vez maior de jogos que são lançados e não atingem o faturamento esperado, o mercado se tornou extremamente volátil para todos que desejam ingressá-lo com uma nova desenvolvedora.

A maioria dos novos empreendedores na área costumam buscar investimento com familiares e amigos, economias pessoais, empréstimos bancários e financiamento através de meios não monetários. Por conta desse cenário, muitos empreendedores precisam lidar com a consequência de que, caso seu primeiro lançamento não tenha o sucesso esperado, não irá existir uma segunda chance, pois toda uma nova leva de investimentos em um negócio que falhou será muito improvável.

Uma ferramenta capaz de ajudar os desenvolvedores a entender qual rumo seu produto está tomando, antes do lançamento, para poder realizar ajustes e de auxiliar investidores a reconhecer os melhores cenários para investir seria benéfica para injetar dinheiro de maneira mais eficiente e ajudar o mercado como um todo a crescer com produtos de mais qualidade.

4.2 Jogos Independentes

Como apresentado previamente na figura Figura 1.1 o crescimento ano a ano do mercado de jogos digitais é notório. Por conta disso, a quantidade de produtoras independentes (popularmente conhecidas como *indies*) vem crescendo exponencialmente. Companhias com equipes pequenas e poucos recursos financeiros sofrem muito para tentar conseguir alguma forma de destaque e falham na maioria das vezes. A publicidade ainda é o principal meio para atrair jogadores, porém para tal é necessário o investimento de recursos que muitas vezes não estão disponíveis para esses pequenos desenvolvedores.

Outro problema de grande ocorrência no universo das produtoras independentes é a estimativa de projeto. Estimar o tempo necessário para a produção de qualquer projeto pode ser difícil, e este é um contratempo muito frequente no mercado dos jogos. Sendo uma indústria que está sempre se inovando é preciso manter o foco no calendário, caso contrário um jogo pode estar sempre recebendo novas ideias e implementações de novas ferramentas pré-lançamento e passar mais tempo em desenvolvimento do que o esperado, perdendo assim a atenção do público.

Os desenvolvedores independentes carecem da falta de informação sobre o verdadeiro potencial de seu próprio produto. Entender quais as características que podem tornar um jogo mais relevante pode potencializar a qualidade dos produtos desenvolvidos e trazer mais confiança tanto para os desenvolvedores quanto para possíveis investidores que estejam na dúvida em relação ao destino e provável retorno de seus investimento.

4.3 Principais Influenciadores

Centenas de jogos são lançados mensalmente, porém apenas alguns conseguem alcançar uma quantidade aceitável de jogadores que justifique o investimento. Dentre os principais fatores capazes de influenciar o sucesso de vendas de um jogo estão:

- Vendas Anteriores: Muitos títulos são sequência de lançamentos anteriores. Os dados de venda de versões passadas pode dar uma ideia do que pode ser esperado do próximo lançamento de uma série.

- Gênero e Desenvolvedora: Todo jogo se encaixa em um gênero e é desenvolvido por alguém, e esse tipo de dado pode ser muito informativo já que tanto desenvolvedoras quanto gêneros de jogos possuem suas bases de fãs que terão interesse no produto.

- Orçamento de Marketing: Na indústria dos games já se tornou uma prática comum empresas investirem quase tanto dinheiro na publicidade de um jogo quanto foi investido no desenvolvimento. Com 90 milhões de vendas e 6 bilhões de faturamento, como indicado pela GamesIndustry [Gam18], *Grand Theft Auto 5* é um jogo que se tornou o produto de entretenimento mais rentável de todos os tempos. O investimento na publicidade pré-lançamento do jogo foi de 150 milhões de dólares, um valor maior do que o custo para desenvolvê-lo. O marketing é uma ferramente fundamental para jogos conseguirem gerar interesse.

- Redes Sociais: Todo tipo de interação pode afetar o interesse em um jogo. Desde análises e artigos até comentários em redes sociais, porém a quantidade de menções não significa que um jogo será um sucesso. É preciso concentrar e analisar todas, tanto positivas quanto negativas.

- Expectativa da Comunidade: O número de fãs do jogo é mais uma medida de sucesso. Analisar que tipos de postagens os usuários colocam nos comentários ou quanto tempo os jogadores tiveram que esperar por um jogo antes de ser lançado. Os fãs da saga *Diablo* esperaram por 12 anos entre o lançamento da segunda e terceira iteração da série. Talvez isso tenha contribuído para o recorde de *Diablo III* para o jogo de PC mais vendido, com mais de 30 milhões de unidades vendidas no total. Por outro lado, esperar muito tempo por um jogo pode diminuir drasticamente suas vendas futuras.

- Ficanciamento Colaborativo: Também conhecido como *Crowdfunding*, um jogo ser financiado nesse molde é fundamental para prever o seu sucesso, dependendo de quantas pessoas o apoiaram e quanto dinheiro foi arrecadado. Às vezes acontece que um jogo é tão envolvente que o feedback positivo dos jogadores superará as expectativas dos desenvolvedores.

Quanto mais fatores forem levados em conta, mais precisas serão as previsões sobre o sucesso do jogo. Além disso, apenas uma pesquisa minuciosa ajudará a estimar os números de estoque corretos. No atual *Estado da Arte* uma ferramenta que considere, detalhadamente, todos os fatores aqui apresentados, ainda não deve ser cogitada, já que são necessários estudos sobre a verdadeira influência que cada um dos pontos apresentados acima exercem sobre o resultado final.

5 Metodologia

Neste capítulo será apresentado como, em uma ótica de obtenção e análise de dados, técnicas de pré-processamento, preparação dos dados e implementação de um algoritmo de aprendizado profundo, visamos resolver o problema contribuindo para o estado da arte.

Este capítulo é dividido em três seções, onde a primeira explica os métodos utilizados na obtenção dos dados para análise. A Segunda faz uma abordagem prática sobre o pré-processamento e preparação destes dados, visando uma melhor acurácia e melhor proveito das características dos dados. E a terceira, finaliza com a exposição da arquitetura e algoritmos utilizados em nosso modelo, bem como sua implementação.

5.1 Conjunto de Dados

Em aplicações de aprendizagem profunda que fazem o uso de dados textuais, é possível utilizar um modelo de *pipeline* padrão [kdn18], como mostrado na Figura 5.1, que consiste em:

1. **Coleta de dados e montagem do conjunto:** Trata-se da obtenção dos dados em si, a partir da mineração de texto por meio de robôs, interfaces de aplicação, entre outros.

2. **Pré-processamento dos dados:** Se trata do processamento dos dados visando a normalização e preparação dos dados não tratados obtidos, para que possam servir como entrada para o classificador em questão. Há vários métodos que podem ser usados nesta etapa, mas a maioria se encaixa nas classes de representação simbólica das palavras, normalização e substituição.

3. **Exploração dos dados e visualização:** Para ter conhecimento sobre do que seus dados se tratam e o seu formato, é importante que se faça um exploração e visualização dos mesmos. Algumas técnicas padrões consistem em fazer uma contagem das palavras e sua distribuição, geração de nuvens de palavras e cálculo de distância entre as mesmas.

4. **Construção do modelo:** Com os dados tratados e normalizados em mãos, e um entendimento sólido sobre os eles, basta construir o modelo propriamente dito para a aplicação em PLN. Este modelo deve incluir um método de seleção de características, modelos de representação de texto, um classificador e/ou um modelo sequencial.

5. **Avaliação do modelo:** É o passo final, onde deve ser feita uma avaliação visando entender se o modelo alcançou o resultado esperado, utilizando métricas e medidas que dependem da aplicação de PLN.

Neste trabalho, os dados foram coletados do site do IGDB [IGD18], a partir do uso de uma interface de programação de aplicações *REST* disponibilizada pelo mesmo. Este tipo de interface permite ao usuário fazer, a partir do protocolo HTTP, operações *GET*, *PUT*, *POST* e

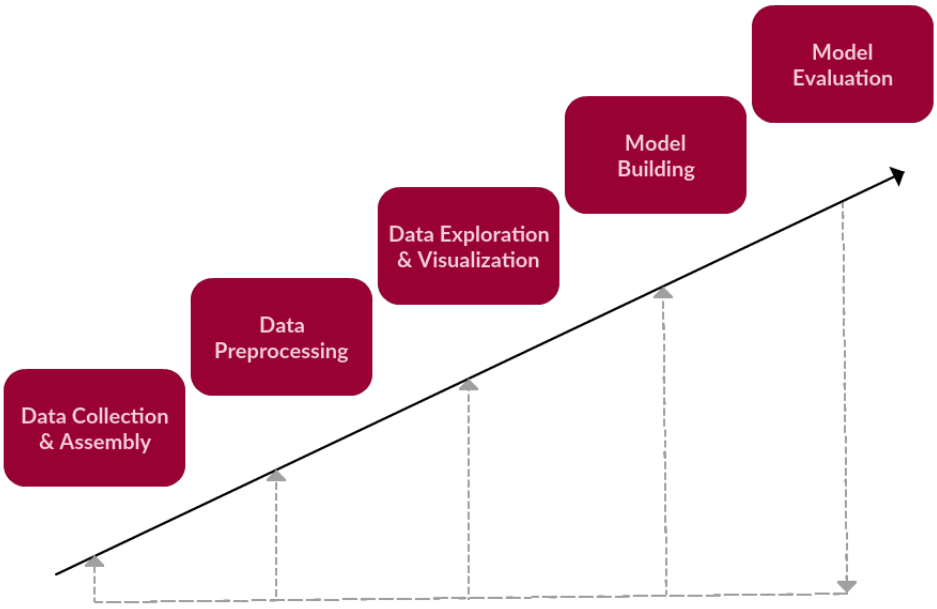


Figura 5.1: Pipeline para implementação de aplicações para PLN.
Fonte: [kdn18]

DELETE à base de dados de uma aplicação. Este tipo de tecnologia permite a realização de consultas utilizando menos banda de dados do que uma abordagem mais robusta como o *Protocolo de Acesso a Objetos Simples*, o que a torna ideal para consultas realizadas através da Internet. Estas interfaces conseguem dividir uma transação em vários módulos de tamanho menor, com cada módulo endereçando uma parte subjacente da transação. Este tipo de modularidade permite aos desenvolvedores maior flexibilidade, porém como a implementação pode ser complicada, diversas empresas privadas que possuem especialidade em armazenamento de dados costumam vender modelos de IPAs *REST*.

O IGDB oferece uma IPA, baseada em *REST*, gratuita para uso comercial ou não comercial. Com *wrappers* oficiais para as linguagens *NodeJS*, *Python*, *Android* e *Java*, essa interface permite a realização de consultas a toda a base de dados do IGDB para desenvolvedores e qualquer usuário interessado.

O *wrapper* utilizado para o consumo desta IPA foi implementado na linguagem de programação *Python*, assim como todo o código resultante deste trabalho. Este envia como parâmetros alguns filtros, como por exemplo o de que os dados em questão devem existir, um limite de dados a serem recuperados a cada requisição (limitando-se a 50), um *offset* que serve como ponteiro para a localização inicial da requisição em questão e os campos que devem ser retornados na requisição, assim como mostrado no código fonte 5.1. Esta IPA retorna os dados em um arquivo estruturado no formato *JSON*, e o formato dos dados recuperados foram conjuntos de jogos com: id, sumário, sinopse, avaliação, perspectiva do jogador, modos de jogo, tema e gênero.

Listing 5.1: Implementação do *wrapper* para consumo da IPA REST do site IGDB.

```
1 import json
2 from igdb_api_python.igdb import igdb
3
4 igdb = igdb("CHAVE_DA_IPA")
5
6 for i in range (0, 100):
7    result = igdb.games({
```

```
8       'filters' :{
9          "[summary][exists]": '',
10         "[storyline][exists]": '',
11         "[genres][exists]": '',
12         "[themes][exists]": '',
13         "[game_modes][exists]": '',
14         "[player_perspectives][exists]": '',
15         "[aggregated_rating][exists]": ''
16      },
17      'limit': 50,
18      'offset': i*50,
19      'order':"date:asc",
20      'fields': [ 'summary',
21                  'storyline',
22                  'genres',
23                  'themes',
24                  'game_modes',
25                  'player_perspectives',
26                  'aggregated_rating'
27              ]
28   })
29   filename = 'datasetigdb.json'
30   with open(filename, 'a') as outfile:
31      json.dump(result.body, outfile)
```

5.1.1 IGDB

O IGDB é uma plataforma de dados que tem como objetivo reunir todas as informações relevantes sobre jogos em um só lugar. A plataforma possui recursos sociais e exploratórios sobre essas informações, além de reunir uma comunidade de jogadores e pessoas da indústria de jogos em um ambiente onde possam se comunicar uns com os outros.

A plataforma oferece informações não censuradas e não manipuladas sobre jogos, sem nenhum tipo de diferenciação ao classificar as opiniões, avaliações ou classificações dos usuários. O status dos desenvolvedores dos jogos é elevado no ambiente, mostrando-os para o usuário, além de impulsionar igualmente jogos tanto independentes quanto de grandes empresas, baseando-se em avaliações e interesse dos usuários.

Um dos princípios do IGDB é a acessibilidade dos dados trazendo seu compartilhamento com todos que possuem interesse em jogos digitais, sejam estes voltados para o desenvolvimento de sites, aplicativos ou diferentes tipos de serviço. Muitas informações disponíveis no IGDB são incluídas pelos próprios usuários, porém todos esses dados são verificados para preservar a integridade de todo o sistema. Assim, as informações inseridas pelo usuário podem contribuir não apenas para a base de dados, mas para qualquer projeto que a utilize.

O site também possui um blog onde seus desenvolvedores publicam entrevistas e ajudam a divulgar jogos. Além disso também são escritos artigos de mais diversos temas (todos relacionados ao mundo dos jogos), onde os autores dissertam sobre assuntos que são de seu interesse, sempre com caráter informativo.

5.2 Pré-processamento dos dados

O pré-processamento de dados é muito importante para as técnicas de aprendizagem profunda. Muitas vezes esses algoritmos não funcionam corretamente em conjuntos de dados

ainda não tratados, pois esses dados podem estar desbalanceados e com ruído. Não existe um algoritmo padrão de tratamento que funcione para qualquer conjunto de dados, sendo assim, uma grande parte do esforço no desenvolvimento de aplicações em aprendizagem profunda é investido no tratamento desses dados. Ainda assim, existem diversas técnicas e algoritmos que podem ser utilizados para este fim.

Ao se obter um conjunto de dados, a primeiro coisa a ser feita é observar os dados em si e atentar-se às suas propriedades. Embora hajam técnicas padrões para o tratamento e normalização dos dados, é possível adaptar e criar novas técnicas personalizadas para um conjunto de dados específico. Um exemplo seria a técnica conhecida como normalização de subtração da média, que consiste em subtrair a média de cada ponto de si mesmo. Esta técnica funciona bem para imagens, por exemplo, mas não é eficaz para uso em texto.

Existem muitas técnicas para o processamento e normalização de dados textuais, porem em sua maioria eles se encaixam em três categorias principais: representação simbólica, substituição e normalização.

5.2.1 Representação simbólica

A representação simbólica, também conhecida como tokenização, é uma técnica que consiste em dividir grandes massas de textos em sentenças menores, ou símbolos (*tokens*). Esta técnica é o primeiro passo para qualquer outro tipo de processamento. Muitas vezes este processo leva o nome de segmentação quando se trata de dividir os dados em partes maiores que palavras, como por exemplo sentenças e frases, e tokenização quando se trata de dividir o texto em palavras.

Neste trabalho foram utilizadas três formas diferentes de segmentação. A primeira, representada no código fonte 5.2, utiliza uma biblioteca para aplicações PLN chamada *SpaCy*[1], seu objetivo é segmentar as sentenças que servirão como entrada para o modelo de aprendizado profundo.

Listing 5.2: Implementação do método de segmentação dos dados.

```
1 import spacy
2
3 NLP = spacy.load('en')
4
5 TEXT = data.Field(sequential=True, tokenize=NLP, lower=True,
6    include_lengths=True, batch_first=True, fix_length=200)
7 LABEL = data.LabelField()
```

A SpaCy é uma biblioteca de PLN popular em *Python*. Ele fornece algoritmos que visam a precisão e velocidade do atual estado da arte, e possui uma comunidade ativa de código aberto. No entanto, como o SpaCy é uma biblioteca relativamente nova de PLN e não é tão amplamente adotado como o NLTK[2]. Ainda não há tutoriais suficientes disponíveis. Porém, ela

[1] spaCy é uma biblioteca para processamento avançado de linguagem natural em Python e Cython. Ele foi desenvolvido com base em pesquisas e foi projetado desde o início para ser usado em produtos reais. A spaCy vem com modelos estatísticos pré-treinados e vetores de palavras, e atualmente suporta tokenização para mais de 30 idiomas. Ele apresenta o analisador sintático mais rápido do mundo, modelos de rede neural convolucional para marcação, análise e reconhecimento de entidades nomeadas e fácil integração de aprendizagem profunda. É um software comercial de código aberto, lançado sob a licença do MIT.

[2] O NLTK é uma plataforma líder para criar programas em Python para trabalhar com dados em linguagem humana. Ele fornece interfaces fáceis de usar para mais de 50 recursos corpora e lexicais como o WordNet, juntamente com um conjunto de bibliotecas de processamento de texto para classificação, tokenização, stemming, tagging, análise e raciocínio semântico, wrappers para bibliotecas NLP de força industrial, e um fórum de discussão ativo. O NLTK é um biblioteca de código aberto líder em aplicações para PLN, tanto simbílicas quanto estatísticas. Ela fornece mais de 50 recursos para o tratamento e manipulação de dados textuais.

é implementada nativamente em vários frameworks de aprendizado profundo, como é o caso do *Pytorch* que foi utilizado na implementação deste trabalho.

A segunda e terceira abordagens de segmentação foram usada para a técnica de geração de dados, como mostra o código fonte 5.3. A segunda consiste em segmentar as massas textuais em sentenças, e a terceira, em palavras. Por uma decisão de projeto, implementação mais simples e um melhor desempenho, foi utilizada para estas tarefas a biblioteca *Python* chamada *Natural Language Toolkit (NLTK)*.

Listing 5.3: Segmentação dos dados para *augmentation*.

```
from nltk import sent_tokenize, word_tokenize

def tokenize_sent(text):
   tokenized = sent_tokenize(text)
   return tokenized

def tokenize_word(text):
   tokenized = word_tokenize(text)
   return tokenized
```

5.2.2 Substituição

A tarefa de substituição se da a partir do cenário em que, após obtidos os dados em formato *JSON*, estes ainda não podem ser enviados ao modelo, pois os modelos de aprendizado profundo, em geral, não lidam bem com dados em formatos de objeto, como é o caso, então a saída é transformar estes dados para um formato tabular, como *CSV*.

Outro ponto, é que neste tipo de arquivo de objetos, quando se tem objetos aninhados, um é apenas referenciado dentro do outro a partir do seu identificador. Um exemplo disso são os gêneros dos jogos, estes são referenciados apenas por seus identificadores únicos dentro do objeto *Game*, então a tarefa a ser realizada neste caso é substituir esses identificadores pelos respectivos valores dos objetos, assim como a atribuição de rótulos às notas com base nos padrões estabelecidos pelo Metacritic [Met18] , como mostrado na tabela 5.1. O código fonte 5.4 expõe a rotina de substituição do conjunto de dados.

Tabela 5.1: Mapeamento das notas em rótulos textuais.

Avaliação	**Nota**
Dislike	0 - 49
Average	50 - 74
Favorable	75 - 89
Acclaim	90 - 100

Listing 5.4: Substituição dos dados.

```
for g in games:

   pps = g['player_perspectives']
   g['player_perspectives'] = []
   for pp in pps:
      for dpp in player_perspectives:
         if dpp['id'] == pp:
            g['player_perspectives'].append(dpp['name'])

```

```
10      gmd = g['game_modes']
11      g['game_modes'] = []
12      for gm in gmd:
13          for dgm in game_modes:
14              if dgm['id'] == gm:
15                  g['game_modes'].append(dgm['name'])
16
17      tem = g['themes']
18      g['themes'] = []
19      for t in tem:
20          for dt in themes:
21              if dt['id'] == t:
22                  g['themes'].append(dt['name'])
23
24      gen = g['genres']
25      g['genres'] = []
26      for gn in gen:
27          for dgn in genres:
28              if dgn['id'] == gn:
29                  g['genres'].append(dgn['name'])
30
31      if (0 <= g['aggregated_rating'] <= 19):
32          g['aggregated_rating'] = 'Dislike'
33      elif (19 <= g['aggregated_rating'] <= 49):
34          g['aggregated_rating'] = 'Unfavorable'
35      elif (49 <= g['aggregated_rating'] <= 74):
36          g['aggregated_rating'] = 'Average'
37      elif (74 <= g['aggregated_rating'] <= 89):
38          g['aggregated_rating'] = 'Favorable'
39      elif (89 <= g['aggregated_rating'] <= 100):
40          g['aggregated_rating'] = 'Acclaim'
41
42  filename = 'datasetlstm.json'
43  with open(filename, 'w+') as outfile:
44      json.dump(games, outfile)
```

5.2.3 Normalização

Antes de passar por outros tipos de processamento, o texto precisa ser normalizado. A fase de normalização é composta de várias sub-tarefas que visam tornar o texto uniforme e diminuir o ruído desnecessário nesta massa de dados. Como mostrado no código fonte 5.5, foram utilizadas as técnicas de normalização de:

- **Remoção de quebras de linha:** Importante para a manipulação de segmentação das sentenças, pois a quebra de linha pode ser reconhecida com o fim de uma sentença ou ser adicionada como ruído ao modelo.
- **Transformação do texto para minúsculo:** Tem o objetivo de manter o texto homogêneo, para evitar que palavras maiúsculas e minúsculas tenham diferentes representações nos vetores de palavras.
- **Remoção de pontuações:** Também utilizada para remover o ruído dos dados, pois após segmentadas as sentenças, as pontuações não se fazem mais necessárias.

- **Remoção de palavras de parada:** São palavras que podem ser consideradas irrelevantes para o método de classificação do modelo. Não existe uma lista universal de palavras de parada usadas por todas as ferramentas de PLN e nem todas ferramentas fazem uso de uma lista dessas palavras. Na língua inglesa, por exemplo, as palavras "*the*", "*and*"e "*a*"podem ser consideradas palavras de parada e também são muito frequentes, e ao remove-las o texto continua tendo o mesmo sentido.

- **Remoção das dez palavras mais frequentes:** Palavras muito frequentes podem adicionar um viés desnecessário ao modelo de classificação. Quando executada, uma tarefa de classificação visa obter a melhor divisão entre os dados e esse tipo de palavra, que ocorre frequentemente em todas as classes, é dispensável para este tipo de tarefa.

- **Remoção das dez palavras menos frequentes:** Ao contrário das palavras muito frequentes que adicionam um viés ao modelo, as menos frequentes podem ser vistas como ruído, pois como aparecem pouco não servem como critério para a classificação.

Listing 5.5: Métodos de normalização dos dados textuais.

```
import pandas as pd
from nltk.corpus import stopwords

stop = stopwords.words('english')

def text_filter(data):
    # Remove quebra de linha
    data["summary"] = \
        data.summary.str.replace("\n", " ")

    # Transforma em lowercase
    data['summary'] = data['summary'].apply(lambda x: " "
        .join(x.lower() for x in str(x).split()))

    # # Remove pontuacao
    data['summary'] = data['summary'].str.replace('[^\w\s]','')

    # Remove stop words
    data['summary'] = data['summary'].apply(lambda x: " "
        .join(x for x in x.split() if x not in stop))

    # Remove palavras mais frequentes
    freq = pd.Series(' '.join(data['summary']).split()).value_counts()[:10]
    freq = list(freq.index)
    data['summary'] = data['summary'].apply(lambda x: " "
        .join(x for x in x.split() if x not in freq))

    # Remove palavras menos frequentes
    freq = pd.Series(' '.join(data['summary']).split()).value_counts()[-10:]
    freq = list(freq.index)
    data['summary'] = data['summary'].apply(lambda x: " "
        .join(x for x in x.split() if x not in freq))

    return data
```

5.3 Geração de dados

Quando se treina um modelo de aprendizado profundo, o que realmente é feito é ajustar seus parâmetros de forma que ele possa mapear uma entrada específica para alguma saída (rótulo). O objetivo da otimização é buscar um ponto ideal em que a perda do modelo é baixa, o que acontece quando seus parâmetros são ajustados corretamente. Se a quantidade de parâmetros do modelo é grande, então a quantidade de exemplos que é preciso fornecer a esta rede também deve ser proporcionalmente grande, para que esse modelo possa obter um bom desempenho.

Para um modelo multi classes, como é o caso, a quantidade de dados para treinamento deve ser suficientemente grande e a quantidade de dados em suas classes deve ser equilibrada, o que não ocorre, como mostra no gráfico da figura 5.2. Neste caso, uma alternativa paliativa é utilizar de métodos de geração de dados, para que assim sejam obtidos dados suficientes para o treinamento do modelo.

Os modelos de aprendizado profundo não são inteligentes. Uma rede mal treinada pode reconhecer que os textos abaixo são completamente distintos:

"A informática me distanciou dos livros, não da leitura."
"Não da leitura, a informática me distanciou dos livros."

As frases acima estão ambas sintaticamente e semanticamente corretas e seu significado é o mesmo, porém um modelo de aprendizado profundo criado para processamento de texto, pode reconhecer essas frases como sendo distintas. Esta técnica é simples, porém muito robusta na forma de geração de novos dados para o treinamento destes modelos, pois gera dados sintéticos que, na maior parte das vezes, retrata o mundo real, sem adicionar ruído aos dados, mantendo o contexto das palavras, o que é muito importante para os algoritmos de vetores de palavras que são utilizados para a criar uma representação destas palavras.

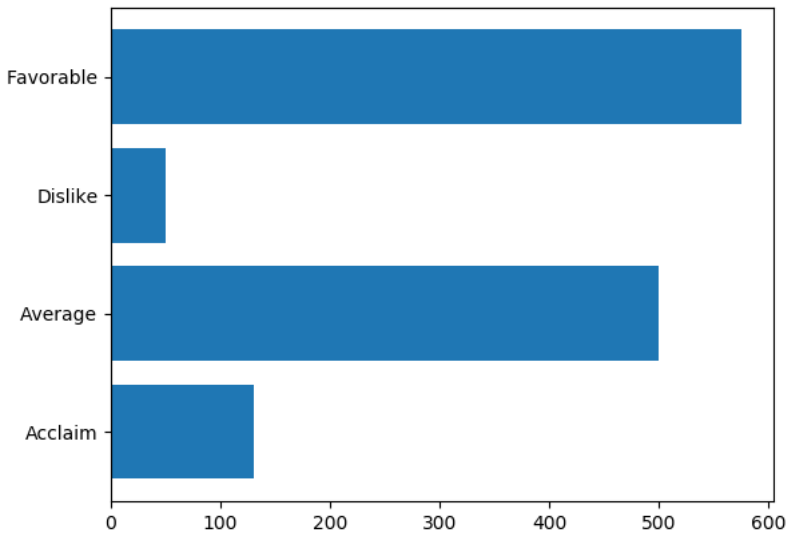


Figura 5.2: Estado do conjunto de dados antes da aplicação das técnicas de geração de dados.

A técnica utilizada para a geração de dados foi basicamente fazer esta mistura de sentenças e palavras em uma frase. Este embaralhamento foi feito a partir de sentenças para classes que já tinham um numero considerável de dados disponíveis, visando sempre manter a integridade dos dados nas questões de ruído e contexto. O embaralhamento das palavras foi

utilizado em classes que continham poucos dados e, neste caso, o objetivo principal da geração de dados foi equilibrar o número de exemplos de cada classe neste conjunto de dados e garantir a integridade quanto a ruídos, sacrificando a integridade quando ao contexto, como mostrado no gráfico da figura 5.3.

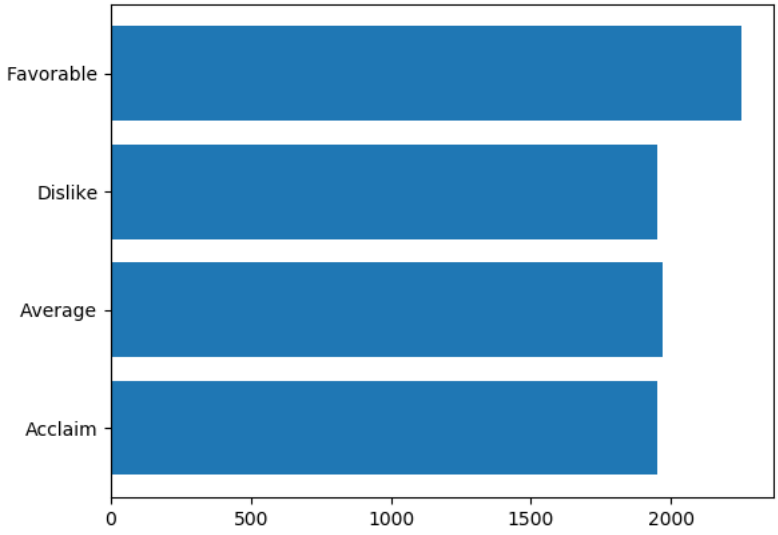


Figura 5.3: Estado do conjunto de dados após a aplicação das técnicas de geração de dados.

Após a geração de dados, uma forma de analisar os resultados obtidos, é exibir a distribuição dos dados de uma forma que possa ser visualizado e analisado, mas os vetores de palavras representados a partir do GloVe [PSM14], no caso deste trabalho, tem 300 dimensões, o que torna inviável a tarefa de visualização de sua distribuição. Uma saída neste caso é utilizar métodos de redução de dimensionalidade.

Este trabalho utiliza um método popular de exploração de dados de alta dimensionalidade chamado t-SNE [vdMH08]. A técnica tornou-se difundida no campo do aprendizado de máquina, uma vez que possui uma grande capacidade de criar mapas bidimensionais a partir de dados com centenas ou até milhares de dimensões. Um exemplo desta representação pode ser observado na Figura 5.4, onde foi selecionada uma pequena região da representação e pode-se observar a proximidade de palavras que estão associadas de alguma forma, como: *fight, combat, warfare, battle*, etc. Este aglomerado de palavras, poderia representar muito bem um jogo que se trata do estilo guerra, FPS, estratégia militar, entre outros.

Na Figura 5.5, é possível observar a distribuição de um pequeno sub-conjunto dos dados, anterior ao processo de geração de dados. Pode-se notar que, no canto superior esquerdo do gráfico, tem-se a representação de algumas datas. Este mesmo sub-conjunto pode ser observado na Figura 5.6, porém desta vez se trata de uma distribuição após a geração dos dados. A partir da comparação dos dois gráficos, é possível chegar a algumas conclusões:

- A distribuição dos dados está melhor, pois temos mais conjuntos com menos dados ou seja, as características que separam os dados ficaram mais marcantes;
- Pode-se perceber que o conjunto de datas citado anteriormente está maior após a geração de dados, o que da a entender que, datas que antes encontravam-se espalhadas pelo espaço de representação, agora estão agrupadas em um só conjunto;

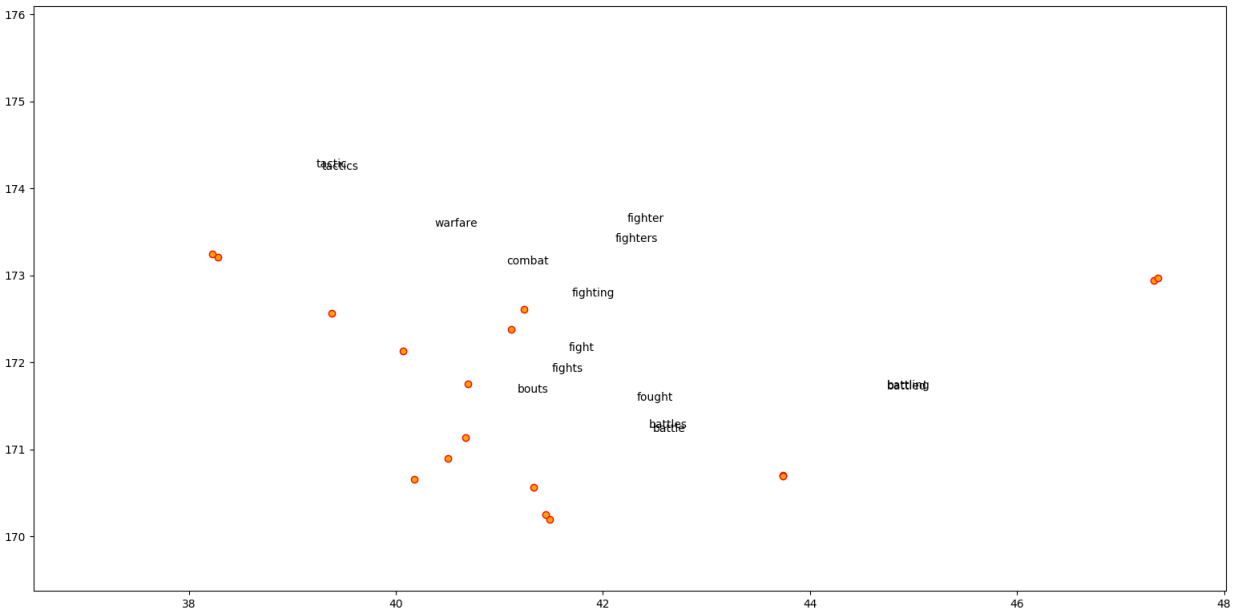


Figura 5.4: Representação dos dados utilizando t-SNE.

- Alguns dados, após a geração de dados, estão sendo representados sequencialmente, como é a questão das datas.

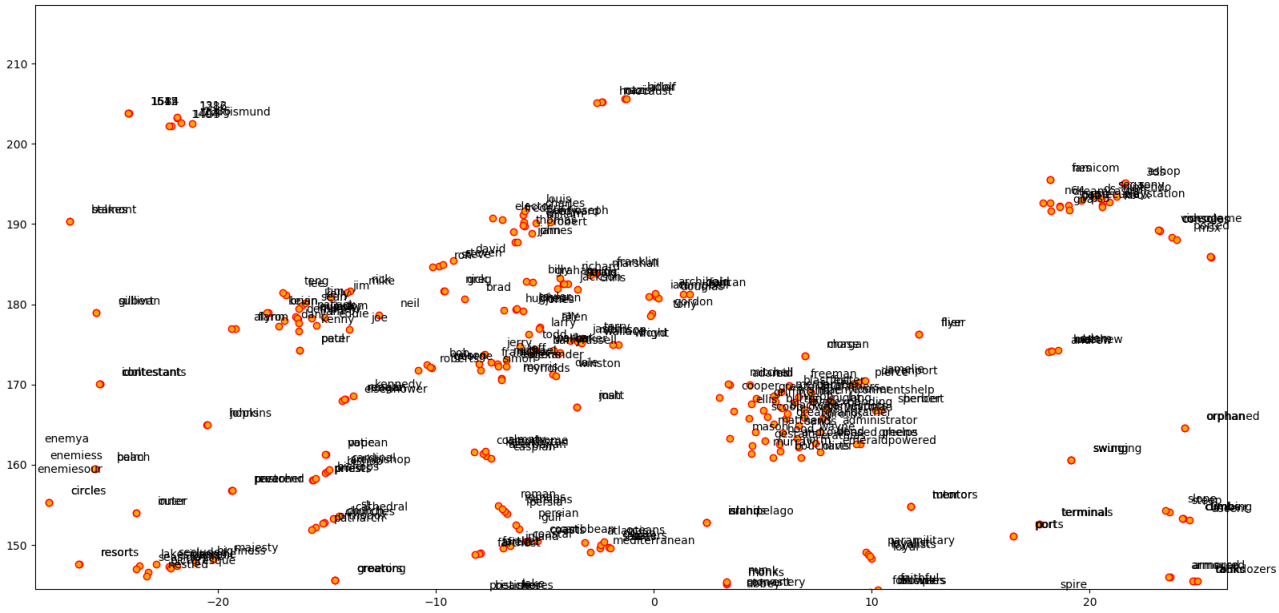


Figura 5.5: Representação anterior geração de dados.

5.4 Modelo

O algoritmo de classificação implementado para este trabalho é um rede neural artificial recorrente, que segue modelos de aprendizado profundo, onde a rede é capaz de lidar com dados de grandes dimensionalidades.

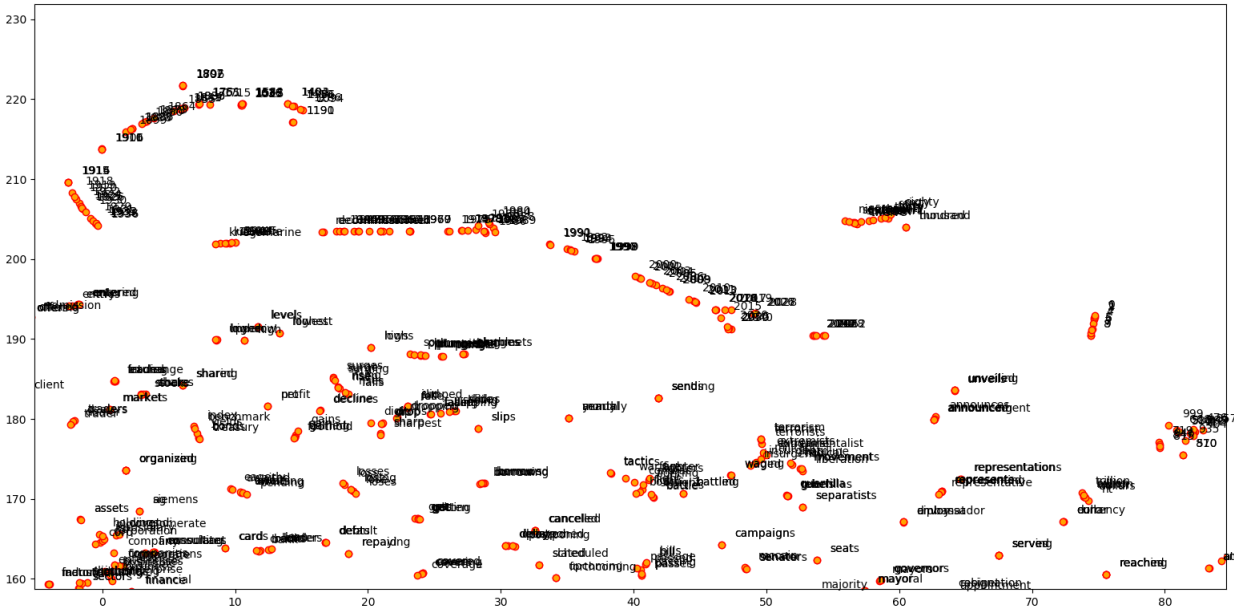


Figura 5.6: Representação após a geração de dados.

O objetivo desta rede é classificar os jogos quanto a suas avaliações, de modo que possa prover um método de predição de aceitação de novos jogos no mercado. Este problema é considerado multi-classes, pois temos ao todo quatro classes a serem classificadas, como mostra a tabela 5.1.

O modelo de rede neural recorrente escolhido foi o de redes LSTM, o qual tem a capacidade de classificar sequências temporais com um bom desempenho, também para conjuntos de dados de grande dimensionalidade. O modelo completo é descrito na figura 5.7.

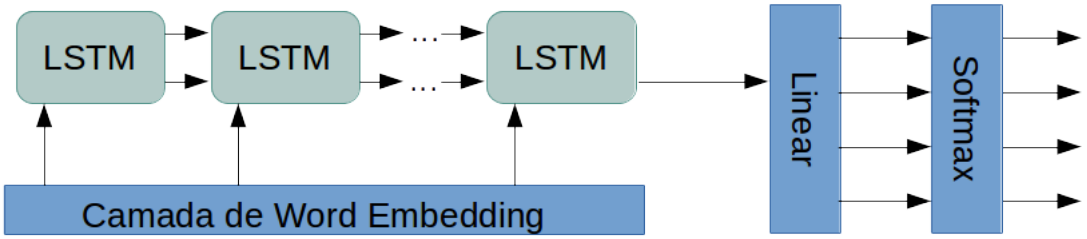


Figura 5.7: Arquitetura do modelo da rede.

Este modelo recebe como entrada um vocabulário que é representado pelo algoritmo GloVe [PSM14]. Este vocabulário, contendo aproximadamente 24.000 símbolos, é a entrada de uma camada de *Word Embedding*[3], o qual tem uma saída de tamanho 300 que se conecta a entrada da camada de células LSTM. Esta camada LSTM contém ao todo duas sub-camadas, onde a primeira contém 300 células LSTM bidirecionais e uma sub-camada de *dropout*, com taxa de 0.5.

A LSTM bidirecional [GFS05] é uma variação da LSTM padrão onde são treinadas duas camadas LSTM por vez. A primeira é treinada seguindo a sequência normal de entrada

[3]O Word Embedding é o nome coletivo de um conjunto de técnicas de modelagem de linguagem e de aprendizado de recursos em processamento de linguagem natural (PLN), em que palavras ou frases do vocabulário são mapeadas para vetores de números reais. Conceitualmente, envolve uma incorporação matemática de um espaço com uma dimensão por palavra para um espaço vetorial contínuo com uma dimensão muito menor.

e a segunda com a sequencia ao inverso. Esta abordagem provê um contexto adicional à rede, resultando em um aprendizado mais rápido e com melhores resultados quanto ao contexto do problema.

O *dropout* [HSK+12], por sua vez, tem o objetivo de regularizar a rede de forma que consiga uma melhor generalização para a classificação. Esta técnica busca simular técnicas de treinamento de múltiplos classificadores, onde são treinados vários classificadores com a mesma instância de dados de treinamento e ao fim são escolhidas as saídas a partir de, por exemplo, métodos de votação ou média. Ele simula esta situação, de forma heurística, a partir da desativação de alguns neurônios da rede, assim seria como se fossem treinadas diferentes redes neurais a cada iteração. Assim, os efeitos dessa eliminação seriam como calcular a média dos resultados de um grande número de redes diferentes. Essas diferentes redes se adaptam de diferentes maneiras, onde em cada camada pode gerar uma melhor generalização, assim diminuindo os efeitos do *overfitting*.

Após passar pela camada LSTM, a dimensão da entrada é reduzida para 128, a qual é tida como entrada de uma camada de neurônios que atribui os rótulos à classificação. E esta camada de neurônios alimenta uma função *softmax*, que transforma as predições em uma distribuição de probabilidades.

6 Resultados Experimentais

Neste capítulo serão apresentados os resultados experimentais sobre o problema de predição de sucesso de jogos digitais apresentado no capítulo 4.

Este capítulo está dividido em duas seções, onde a primeira se trada dos resultados experimentais do modelo de predição exposto na sessão 5.4. E na segunda temos uma discussão sobre os resultados obtidos.

Todos os experimentos foram realizados em um computador Dual-Core Intel® Core™ i3-2100 CPU @ 3.10GHz com 3MB de memória cache, 12GB de memória RAM e uma NVIDIA GeForce GTX 1060 de 3 GB com 1152 núcleos.

6.1 Treinamento e classificação

O modelo descrito na sessão 5.4 foi treinada em um total de dez épocas, com mini-lotes de dados de tamanho 32.

Para estimar a taxa de perda da rede, foi utilizada a função de perda de entropia cruzada[1], que combina os métodos de *negative log likelihood loss*, útil na classificação de problemas multi-classe, e *log softmax*, que é basicamente o logaritmo da função de *softmax* para uma entrada n-dimensional. Esta função é descrita na equação 6.1, onde x é o resultado da classificação e $class$ é o conjunto de n-classes.

$$loss(x, class) = -log\left(\frac{exp(x[class])}{\sum_j exp(x[j])}\right) = x[class] + log\left(\sum_j exp(x[j])\right) \tag{6.1}$$

A função de otimização utilizada na retro propagação foi a Adam [KB14], que se trata de um algoritmo para otimização baseada em gradiente de primeira ordem de funções objetivas estocásticas, com uma taxa de aprendizagem de 0,001.

A partir dos experimentos de classificação, foram calculadas algumas métricas para avaliar a capacidade de acerto e o desempenho do modelo: Acurácia, precisão, revocação e medida F1.

Estas métricas levam em consideração o número de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos.

A acurácia 6.2, leva em consideração o número de entradas classificadas corretamente sobre o número total de classificações.

$$Acuracia = \frac{tp + tn}{tp + tn + fp + fn} \tag{6.2}$$

[1] A perda de entropia cruzada mede o desempenho de um modelo de classificação cuja saída é um valor de probabilidade entre 0 e 1. A perda de entropia cruzada aumenta à medida que a probabilidade prevista diverge do rótulo real. Um modelo perfeito teria uma perda de log de 0.

A precisão 6.3, considera a taxa de acerto quanto ao número de entradas classificadas corretamente em uma classe, ou seja, o número de verdadeiros positivos sobre a quantidade total de positivos, ou o inverso no caso da classe de negativos.

$$Precisao = \frac{tp}{tp + fp} \tag{6.3}$$

A revocação 6.4 calcula a frequência com que o classificador encontra os exemplos de uma classe, ou seja, o quão frequente uma classe é classificada corretamente.

$$Revocacao = \frac{tp}{tp + fn} \tag{6.4}$$

A medida F1 6.5 é uma métrica que calcula a média harmônica entre precisão e revocação de modo a trazer um número único que indique a qualidade geral do modelo.

$$F1 = 2\frac{Precisao * Revocacao}{Precisao + Revocacao} \tag{6.5}$$

Os resultados obtidos foram separados em dois conjuntos, os resultados antes e os depois da geração de dados descrita no na sessão 5.3, para que seja possível analisar os resultados deste método e como modelo proposto reage ao mesmo. Na tabela 6.1 há a relação das métricas entre os experimentos e nas figuras 6.1 e 6.2 são expostos os resultados de forma a serem representados por matrizes de confusão exibindo os resultados dos experimentos sem e com a geração de dados, respectivamente.

Tabela 6.1: Cálculo das métrica nos casos de teste.

Dados	**Acurácia**	**Precisão**	**Revocação**	**Medida F1**
Antes da geração	0,55	0,56	0,55	0,55
Após a geração	0,95	0,95	0,95	0,95

6.2 Considerações

Analisando os resultados obtidos nos experimentos realizados podemos concluir que, considerando o tamanho do conjunto de dados inicial, os resultados foram satisfatórios tanto antes quanto depois da geração de dados. Isso se da pois, utilizando a divisão proposta para o conjunto de 60% dos dados para treino, 20% para validação e 20% para teste, temos que mesmo com o conjunto de treino pequeno, foi alcançada uma acurácia de 55% na base de testes, o que pode ser considerado aceitável pelo fato de se tratarem de aproximadamente 25.000 símbolos sendo classificados quanto ao seu contexto.

Após a geração de dados os resultados obtidos ultrapassaram as expectativas, com acurácia acima de 95%, mas devemos analisá-los com um certo cuidado. Os dados gerados sinteticamente preservam e reforçam a questão do contexto em que os dados estão inseridos, mas não injeta novos símbolos na representação. O vocabulário continua sendo o mesmo, portanto para dados fora do contexto deste conjunto de dados, o resultado não será igual.

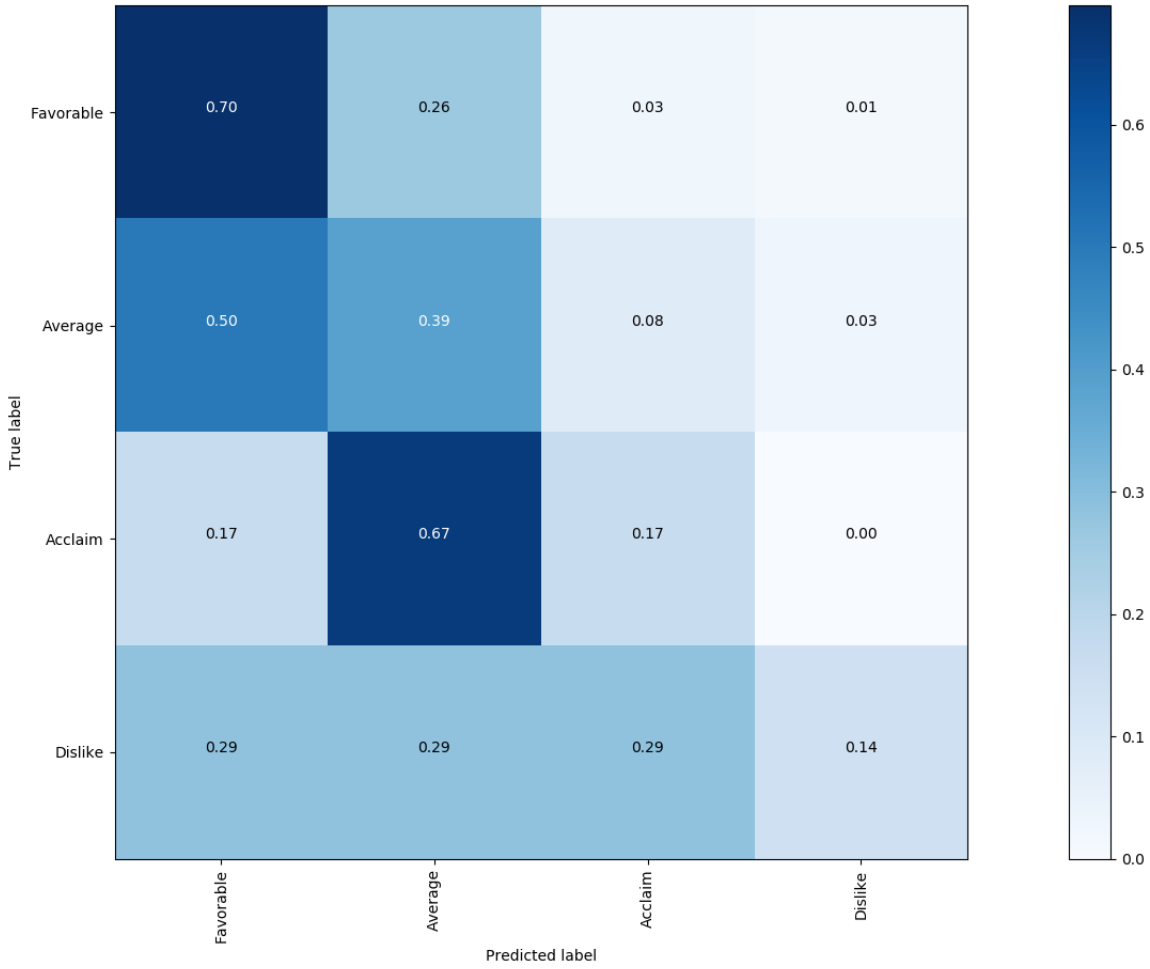


Figura 6.1: Matriz de confusão resultante dos experimentos anteriores a geração de dados.

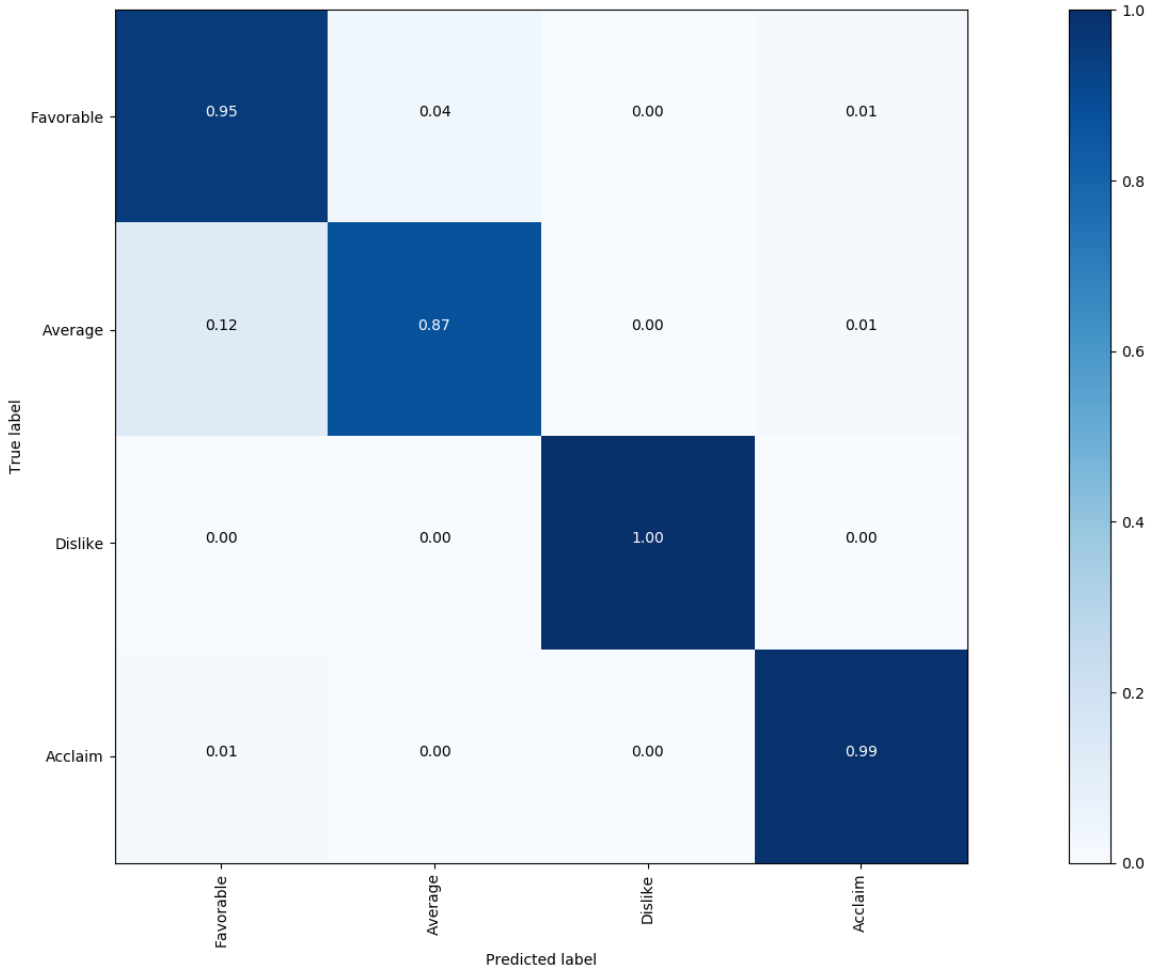


Figura 6.2: Matriz de confusão resultante dos experimentos posteriores a geração de dados.

7 Conclusão

Este trabalho teve como premissa propor um modelo de previsão de desempenho em avaliação de jogos digitais, utilizando como base modelos de redes LSTM. Foi realizada uma série de experimentos empíricos utilizando dados reais da Internet Game Database [IGD18], e analisados os resultados. O modelo proposto nesse trabalho tem resultados melhores que outros trabalhos desenvolvidos com o mesmo objetivo. Os resultados podem ser considerados promissores já que foi comprovada a capacidade de previsão quando comparada com outras abordagens apresentadas na literatura. Mesmo com o tamanho da entrada não sendo grande, o algoritmo demonstrou uma grande capacidade de aprender a partir dela.

Na seção 7.1 são analisadas as contribuições realizadas e em seguida em 7.2 é feito um balanço sobre os potenciais trabalhos futuros na área.

7.1 Contribuições realizadas

Foi projetado um modelo de previsão de desempenho em avaliação de jogos digitais utilizando uma base de dados que reúne informações imparciais, sobre mais de mil jogos, fazendo uso de um algoritmo de redes neurais recorrentes. Além disso, foi validado o modelo através de experimentos e apresentamos uma análise de seu desempenho. Foram alcançados resultados satisfatórios quanto ao *Estado da Arte*, o que abre espaço para uma nova variedade de pesquisas na área.

Foram alcançados resultados superiores aos outros trabalhos na área com a implementação de um LSTM bidirecional, unificado a geração de dados que foi utilizada para equilibrar o número de exemplos de cada classe do conjunto, garantindo integridade quanto a ruídos.

Foi proposto um modelo genérico de rede LSTM, que pode ser utilizado para outras aplicações de classificação de dados de sequências temporais.

7.2 Trabalhos futuros

O campo de estudos tanto de avaliações quanto de vendas de jogos ainda oferece muitas possibilidades de estudo, portanto, esse trabalho pode ser complementado e aprimorado de diversas formas.

Pode ser que existam outros atributos que ajudem a melhorar os resultados e que podem ser estudados como, por exemplo, correlações entre o sucesso dos jogos com comentários em redes sociais, influência de notícias e artigos em sites especializados, cliques em ferramentas de busca na web ou visualizações em serviços de vídeo. O uso de outras bases de dados especializadas nos tipos de atributos mencionados é um passo fundamental para a validação do modelo proposto.

É possível que alguns dos atributos utilizados não possuam nenhum tipo de efeito nos resultados e uma análise, individual, desses atributos pode determinar isso.

Do ponto de vista financeiro, ainda é preciso realizar estudos sobre quais são as melhores estratégias de investimento como base nos dados que podem ser obtidos a partir dos elementos aqui apresentados.

Também fica proposto o estudo do trabalho *Extraindo Características De Jogos Utilizando Redes Neurais Artificiais*, que foi criado em conjunto com este projeto, já visando um possível unificação para uma única plataforma.

Referências

[BJGJ96] Thomas J Bergin Jr and Richard G Gibson Jr. *History of programming languages—II*. ACM, 1996.

[Cho56] Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.

[CPH08] D Manning Christopher, Raghavan Prabhakar, and Schutza Hinrich. Introduction to information retrieval. *An Introduction To Information Retrieval*, 151(177):5, 2008.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[D.05] Klein D. Cs 294-5: Statistical natural language processing. `http://www.cs.berkeley.edu/wklein/cs294-5`, 2005. Acessado em 10/10/2018.

[DDM03] Edwin D. Reilly Daniel D. McCracken. Backus-naur form (bnf). *Encyclopedia of Computer Science*, 4:129–131, 2003.

[Ehr11] Steven Emil Ehrenfeld. Predicting video game sales using an analysis of internet message board discussions. Master's thesis, A Thesis Presented to the Faculty of San Diego State University, San Diego - CA, April 2011.

[Gam18] GamesIndustry.biz. Gta v is the most profitable entertainment product of all time. `https://www.gamesindustry.biz/articles/2018-04-09-gta-v-is-the-most-profitable-entertainment-product-of-all-time`, 2018. Acessado em 27/11/2018.

[GFS05] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ICANN'05, pages 799–804, Berlin, Heidelberg, 2005. Springer-Verlag.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Hoc98] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[HS97] LecunRSepp HochreiteR and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[HSK+12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[IGD18] IGDB. Internet game database. `https://www.igdb.com`, 2018.

[K+95] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[kdn18] kdnuggets.com. A general approach to preprocessing text data. `https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html`, 2018. Acessado em 28/11/2018.

[KL80] Virginia Klema and Alan Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LFL98] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[LLS+15] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Met18] Metacritic. How we create the metascore magic. `https://www.metacritic.com/about-metascores`, 2018. Acessado em 01/12/2018.

[New18] Newzoo. Mobile revenues account for more than 50% of the global games market as it reaches $137.9 billion in 2018. `https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/`, 2018. Acessado em 26/11/2018.

[PMN11] Wendy W Chapman Prakash M Nadkarni, Lucila Ohno-Machado. Natural language processing: an introduction. *J Am Med Inform Assoc*, 18:544–551, 2011.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[Tri13] Craig Trim. What is language modeling. *April 26th*, 2013.

[Trn17] Michal Trněný. Machine learning for predicting success of video games. Master's thesis, Masaryk University Faculty of Informatics, Czechia, Spring 2017.

[Tur09] Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[Wer90] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.