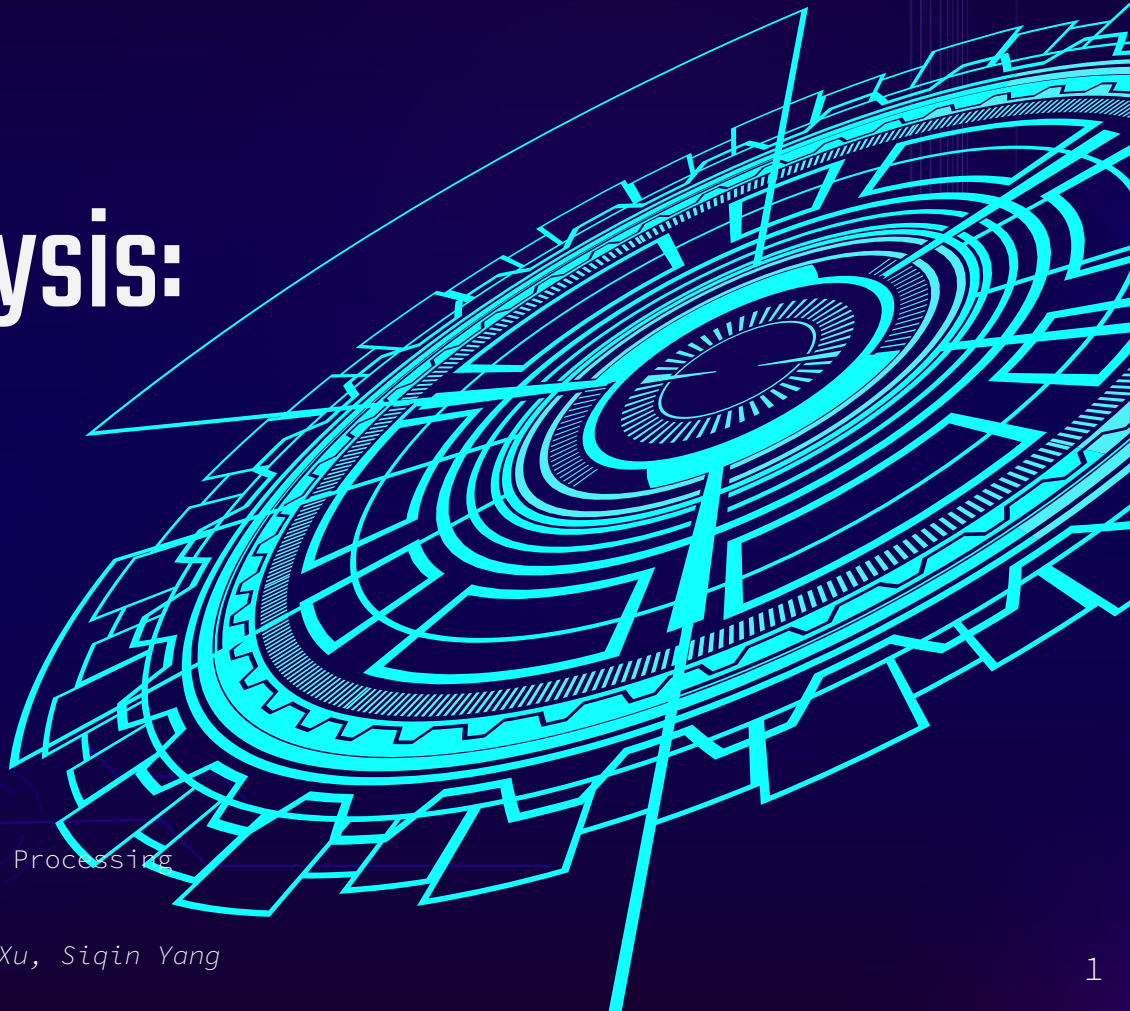
Sentiment Analysis: Steam Reviews



DSO560: Text Analytics & Natural Language Processing
Group Project

Presented By *Yi Gao, Ningxi Wang, Lingyi Xu, Siqin Yang*

TABLE OF CONTENTS

01

Problem Statement

Company Introduction
Team Introduction
Business Use Cases

Company Introduction

Steam is the largest digital distribution platform for PC gaming. Our services include game purchasing, installation, and updates, along with player community, mods workshop, etc.

We operate on a **marketplace business model**, connecting supply and demand. We make the majority of money through **commission**, taking 20-30% cut for every game sold. Other revenue streams include fees for developer tools, selling hardwares, and licensing fees.

Team Introduction

As **data scientists** at Steam, we develop AI solutions for operation optimization.

Assumptions: we have millions of reviews for the 50,000+ games on our platform. However, there is **no clear signal** whether the review creator likes the game or not.

To extract information from these reviews and conclude as their sentiment for game recommendation, we want to build solid models with our 70,000 **manually-labeled** reviews, each with a tag “if recommended” generated by human.

Problem Statement

Data

The original Steam review dataset is from Kaggle.com[1]. We sampled it for our analysis.
Dimension: (70,000, 6)
Fields: record_id, app_id, app_name, review_id, review, recommended
Ratio of recommended : not recommended = 1 : 1

Problem #1

There are many reviews generated every second. We need to have a pipeline that replaces the manual process of deciding whether a review is in favor of a game.

Business Use Case #1

A sentiment prediction model trained on manually-labeled data can automatically classify any new review it sees in the future. We can then feed the predicted sentiment into an abundance of downstream analyses.

Problem #2

The absence of sentiment/preference information has been limiting our ability to capture gaming trend and to optimize the offerings on the platform.

Business Use Case #2

Once we know a user's sentiment towards a game based on the review, we can study the attributes/elements/topics that cause people to like/dislike a game. This helps with decisions w.r.t. our game listing portfolio and deals with game developers.

1. https://www.kaggle.com/datasets/najzeko/steam-reviews-2021

02 Sentiment Prediction

BOW: Logistic Regression, MLP
Sequential: RNN, LSTM, Transformer

DATA PREPROCESS

Text preprocessing is one of the most important tasks in Natural Language Processing.

RegEx Cleaning

Apply **Regular Expressions** & **textacy** package to clean text

Stopwords Removal

Remove stopwords by **spaCy**'s pretrained **en_core_web_sm** model

Lemmatization

Using **NLTK WordNetLemmatizer** to group together the inflected forms of a word

Sentiment Prediction Models

Bag-of-words Models

In these models, a text is represented as the bag of its words, disregarding grammar and even word order but keeping multiplicity.

Logistic Regression

Baseline classification model using Logistic function to model the conditional probability of target

Multilayer Perceptron

A fully connected class of feedforward artificial neural network

Sequential Models

In these models, a text is taken into account by word order, weighting highly on its context.

Recurrent Neural Network

A type of neural network that deals with sequential data, using both the present and the recent past as inputs

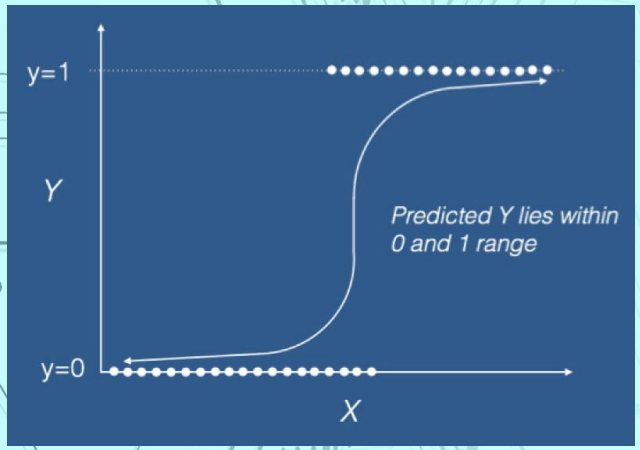
Long Short-term Memory Network

An advanced RNN capable of learning long-term dependencies and thus works better for long reviews

Transformer

A deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data

Logistics Regression



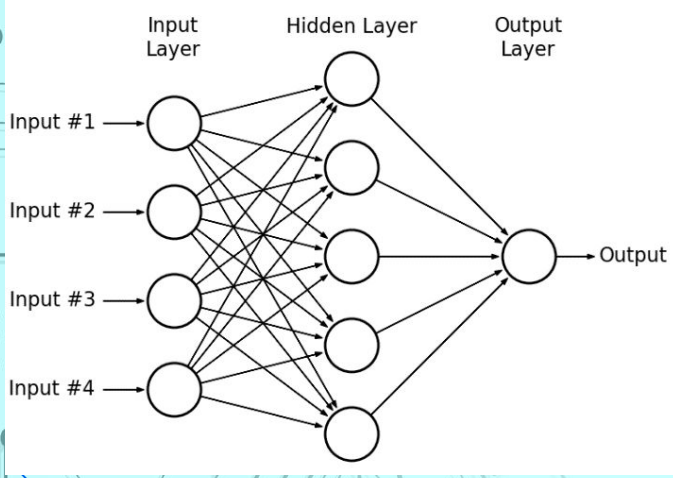
- Baseline classification model
- Using Logistic function to model the conditional probability of target

We tried 3 types of representations of text:

Text Representation	TF-IDF (unigram)	TF-IDF (bigram)	self embedded
Sample Features	abandon ability able	able get absolutely love access game	vectors that are not explainable
Training Accuracy	83.0%	71.7%	59.1%
Testing Accuracy	81.1%	71.1%	58.5%

Diagram from https://www.analyticsvidhya.com/blog/2021/07/perform-logistic-regression-with-pytorch-seamlessly/

Multilayer Perceptron

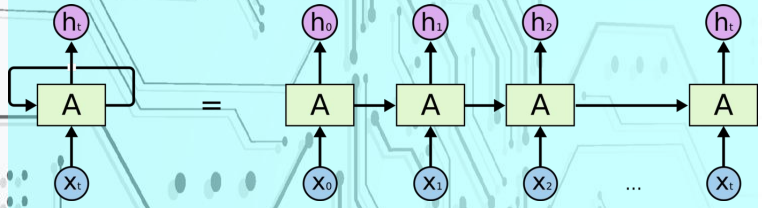


- A MLP network maps an input vector to an output scalar, or typically a vector of axes into a single dependent variable.
- Our MLP models consist of an input layer, an embedded layer, one hidden layers and an output layer.

Embedding Dimension	128	64	256
Hidden Dimension	64	64	64
Training Accuracy	85.8%	75.6%	87.3%
Testing Accuracy	76.1%	74.1%	75.2%

Diagram from
https://www.researchgate.net/figure/A-hypothetical-example-of-Multilayer-Perceptron-Network_fig4_303875065

Recurrent Neural Network

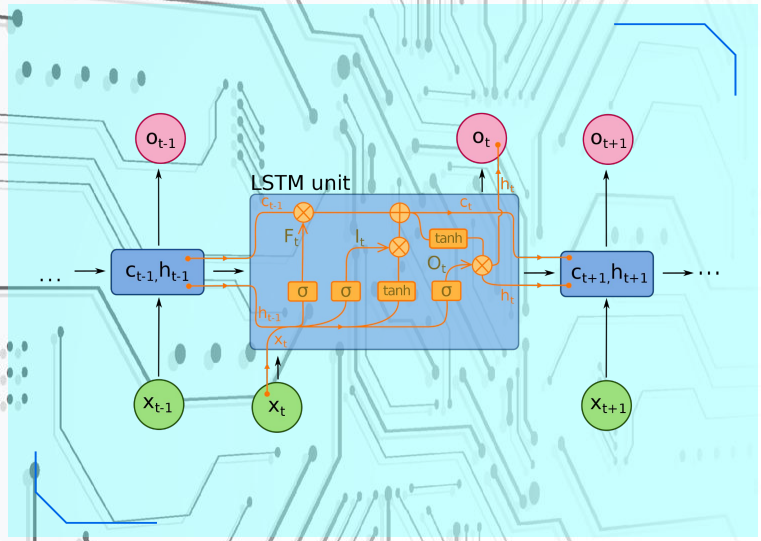


- A type of neural network that deals with sequential data, using both the present and the recent past as inputs.
- Our RNN models make use of GloVe embeddings.

Max Sequence Length	150	150	200
Training Epochs	20	9	10
Training Accuracy	78.7%	79.3%	80.8%
Testing Accuracy	80.1%	77.7%	78.0%

Diagram from https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3

Long Short-term Memory Network

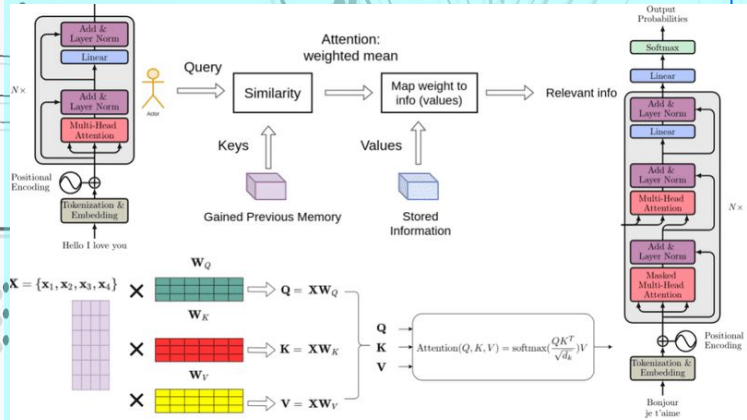


- An advanced RNN capable of learning long-term dependencies and thus works better for long reviews
- Our LSTM models make use of GloVe embeddings.

Max Sequence Length	150	300	1000
Training Epochs	12	15	5
Training Accuracy	86.3%	87.2%	83.7%
Testing Accuracy	81.4%	81.0%	84.9%

Diagram from https://en.wikipedia.org/wiki/Recurrent_neural_network

Transformer



- A deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data
- We use Hugging Face pre-trained distilbert-base-uncased-finetuned-sst-2-english model[1]

Overall Accuracy	81.4%

1. https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english

Diagram from https://theaisummer.com/transformer/

Model Results

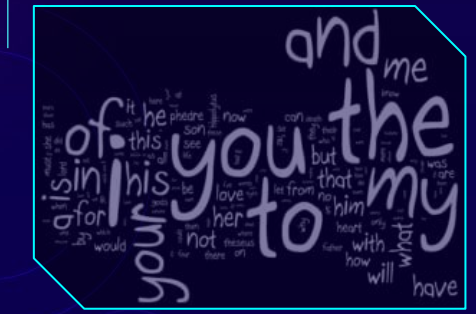
Model Type	Model		Parameters		Performance	
Bag-of-words Models	Logitic Regression	Features	penalty	solver	training	testing
		TF-IDF (unigram)	l2	lbfgs	83.0%	81.1%
		TF-IDF (bigram)		liblinear	71.7%	71.1%
		self-embedded		liblinear	59.1%	58.5%
	MLP	Features	EMBEDDING_DIM	HIDDEN_DIM	training	testing
		self-embedded	128	64	85.8%	75.6%
			64	64	75.6%	74.1%
			256	64	87.3%	75.2%
Sequential Models	RNN	Features	MAX_SEQUENCE_LENGTH	epochs	training	testing
		GloVE vectors	150	20	78.7%	80.1%
			150	9	79.3%	77.7%
			200	10	80.8%	78.0%
	LSTM	Features	MAX_SEQUENCE_LENGTH	epochs	training	testing
		GloVE vectors	150	12	86.3%	81.4%
			300	15	87.2%	81.0%
			1000	5	83.7%	84.9%
	Transformer	Model			overall	
		distilbert-base-uncased-finetuned-sst-2-english model			81.4%	

Among our models, the LSTM model, using GloVe embedding features with 1000 Max Sequence Length and 5 training epochs yields highest testing accuracy.

03 User Preferences Analysis

Key Features of User Preferences

Step 1: Remove Stopwords

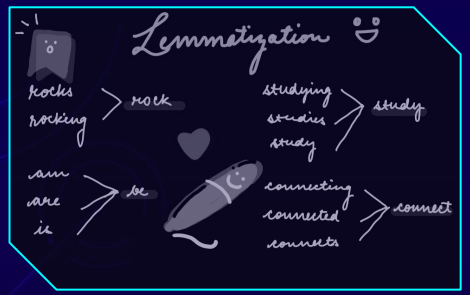


Stopwords are available in abundance in any human language. By removing these words, we remove the low-level information from our text in order to give more focus to the important information.

Removal of stop words definitely reduces the dataset size and thus reduces the training time due to the fewer number of tokens involved in the training.

- Removed non-English characters
- Removed common stopwords in English
  - nltk library default
- Removed consecutively repeated patterns
  - [re.sub(r'^(.+?)\1+', r'\1', word)
- Removed customized stopwords based on business use cases
  - most common words that are too general (based on word count)
  - adverbs that emphasis certain verbs

Step 2: Lemmatization



- Lemmatization is the way to remove inflectional endings only and to return the base or dictionary form of a word by doing things properly with the use of a vocabulary and morphological analysis of words.
- Stemming is the other way to chops off the ends of words in the hope of achieving this goal correctly, and often includes the removal of derivational affixes.
- In this case, We choose **lemmatization** instead of stemming since we care about the context of each token.

Step 3: Regex Cleaning

Regular expressions (regex) are essentially text patterns that we can use to automate searching through and replacing elements within strings of text. Using the regex method is a faster way of cleaning text data that is simpler and quicker to use compared to manually splitting the strings.

Some Regex cleaning examples:

- Substitute words like “don’t”, “didn’t”, “won’t”, etc. with “not”
- Substitute “ive” with “i have”
- Substitute “youve” with “you have”
- Substitute “id” would “i would”
- Change words like “nooooooo”, “goooood”, “baddddddddd” into their original forms
- Group synonyms (“great”, “best”, “amazing” -> “good”

Step 4: TF-IDF Vectorizer

TfidfVectorizer weights the word counts by a measure of how often they appear in the documents.It helps us in dealing with most frequent words by penalizing them.

- We tried n-grams of 2, 3, and 4 to determine which would be the most suitable n-gram for our analysis.
- We used for loop and there is the filter for TF-IDF vectorizer
  - ngram_range=(2,2)/ (3,3)/ (4,4)
  - binary=True: repetition in a comment counts only once
  - max_features=1000: take the top 1000 tokens
  - token_pattern=r'\b[a-zA-Z]{3,}\b': only take english string that longer than three
  - stop_words=stopwords: remove stopword

Reasons Why A Game Is **LIKED**

Extract from 20 most descriptive tokens using TF-IDF Vectorizer

The most common themes in good reviews that can be addressed with different teams are:

Operations, Sales and Marketing Teams

Worth the Price

- The price spent on the games are well-worthy and users would recommend these games to others.

Game Design Teams

Multiplayer Experience

- Users enjoyed the multiplayer experience and play with friends for online contents.

Game Content

Games with the following characteristics are more welcomed among users:

- Good art style
- Curated storyline

Game Features

User preferred game features:

- Open world
- Early Access
- Turn base strategy

Reasons Why A Game Is **DISLIKED**

Extract from 20 most descriptive tokens using TF-IDF Vectorizer

The most common themes in poor reviews that can be addressed with different teams are:

Operations, Sales and Marketing Teams

Age & Region Block

Some games might have area and age restrictions that could negatively impact user experience.

Not Worth the Money

The quality or playfulness of some games make user think the games do not worth the money spent.

Game Design Teams

Single Player Mode

Single player mode with minimal online content in some games are not ideal for users who want to play with friends.

Game Play Experience

Some complains:

- Games are slow to load
- Too many bugs
- Not having many new content

04 Business Applications

ROI Estimation

Return on Investment Estimation

With the successful implementation of our project, we expect **2 major revenue streams**:
1. Design and sell **AI data products** to game developers, which will facilitate their production
2. Negotiate **revenue share** with game developers, based on understanding of user preference

AI data products

Leveraging our reviews & user preference analysis, build prepackaged and customized data solutions that enable developers to make better games

$1,000 for prepackaged x 50 sold/month +
$3,000 for customized x 5 sold/month

$65,000 Monthly return

Negotiation of revenue share

~1,000 new games released/month
We anticipate 5%, or 50, likely to be popular, based on the alignment of their features with Steam user preference
We negotiate a 5% lift in commission rate

50 games x 1,000 purchases/game/month x $30 avg price x **5%** additional commission

$75,000 Monthly return

$35,200 Monthly cost

Salary of data scientists updating models w/ new reviews:
$90/h x 80 hours = **$7,200**
Salary of business analysts generating new insights from DS outputs:
$60/h x 200 hours = **$12,000**
Salary of product managers compiling & commercializing recommendations & solutions:
$80/h x 200 hours = **$16,000**

Return on Investment = ($65,000 + $75,000) / $35,200 = 398%


Thanks for Listening!