Applications of AI/ML

Examples from Industry use cases

Raman Sankaran

Agenda

- ML project lifecycle
- Example 1: Spam detection
- Example 2: Predicting viral sequences
- Example 3: Entity resolution

Applying ML to industrial use cases

- Identify specific business problem, which may benefit from ML
    - No use of ML when simpler deterministic solutions work
    - How often the predictions needed: online vs batch
- What is the specific ML problem:
    - Classification / clustering / ranking / …
    - How correlated is this to the business problem
- Data
    - Is it easy to collect data for training ?
    - Manual collection vs self supervised
    - How to store them efficiently for processing
- Choose models depending on the requirements identified
    - Fast serving may benefit from lighter models
- Serving infrastructures
    - Streaming (Kafka, Flint) or batch
- Continuous Monitoring
- Metrics

Machine learning Project Lifecycle

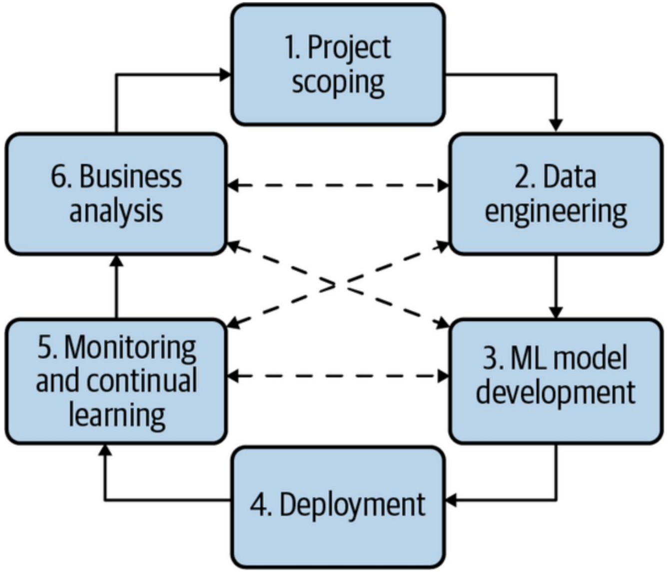


Image from “Designing Machine Learning Systems”, Chip Huyen

Spam Detection

Problem Setup

Context: Social networks, email, user generated content.

Multiple types

- Money Scam
- Misinformation / Fake information
- Commercial spam
- Hateful content
- …

Problem: Identify if a given content is spam

- Impossible to scale any manual solution to large scale
- Need real time prediction

Text Spam identification

Challenges

- Variable length input
    - Most ML classifiers are designed for fixed length features.
        - LR, SVM: d-dim
    - How to encode the input into fixed length information ?
- Multiple variants of same word, typos, …
- Generic words (stopwords)
- Labelling samples
    - Limited by available manual labelers.
    - Low spam rate in practice

Discussion: approaches ?

Prediction protocol: example

Example

Dear Raman, Bank of America is closing your bank account. Please confirm your PIN at [link] to keep your account activated.

1. *Tokenize*

[‘Dear’, ‘Bank’, ‘of’, ‘America’, ‘is’, ‘closing’, ‘your’, ‘bank’, ‘account’, ‘Please’, ‘confirm’, ‘your’, ‘PIN’, ‘at’, ‘to’, ‘keep’, ‘your’, ‘account’, ‘activated’]

2. Remove Stopwords

[‘Dear’, ‘Bank’, ‘America’, ‘closing’, ‘your’, ‘bank’, ‘account’, ‘Please’, ‘confirm’, ‘your’, ‘PIN’, ‘keep’, ‘your’, ‘account’, ‘activated’]

3. Stemming

['dear', 'bank', 'america', 'close', 'your', 'bank', 'account', 'pleas', 'confirm', 'your', 'pin', 'keep', 'your', 'account', 'activ']

Prediction protocol: example

4. Count features

{'dear': 1, 'bank': 2, 'america': 1, 'close': 1, 'your': 3, 'account': 2, 'pleas': 1, 'confirm': 1, 'pin': 1, 'keep': 1, 'activ': 1}

6. Predict the score using trained model

7. Arrive at the label using the score.

- Use the operating threshold for the desired precision level.

Model training

Data - list of labelled examples (spam / non-spam)

Preprocessing - as discussed in the last example

- Tokenizing, remove stopwords, stemming, etc.

Featurizing

- Construct vocabulary
- Build count vectors

Model

- Learn binary classifiers (LR, SVM, Trees, NNs, etc.)

Improving the model

Spatial correlation

- N grams
    - Concatenate nearby word tokens
    - Vocabulary increases with N.

Distinguishing generic vs specific words

- Scale the counts by a weight
- Inverse document frequency

Illustration - <See Notebook>

Practical considerations

Operating thresholds

- Typical requirements on precision (>90%)
- Also depends on relative importance between recall and precision
- Need to tune the threshold level accordingly

Data drift

- The samples distribution may change with time
- Need to retrain the model in regular intervals

Modern NLP

Word Representations (Embeddings)

- Word2Vec, Glove, d = 300
- f('Man) - f('woman') = f('king') - f('queen')

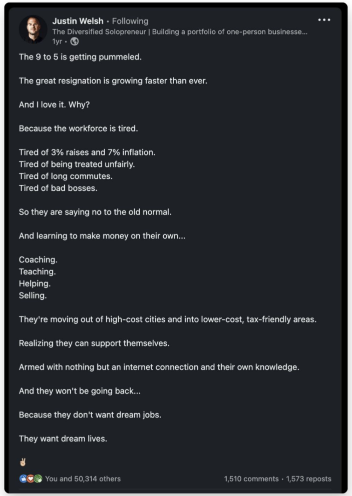
Exploit the temporal nature

- Recurrent neural nets
- RNN, LSTM, GRU, etc.

State-of-the-art

- Transformers based: BERT, GPT, LLaMa, LaMDA

Predict Viral Sequences



Justin Welsh · Following
The Diversified Solopreneur | Building a portfolio of one-person businesse...
1yr ·

The 9 to 5 is getting pummeled.

The great resignation is growing faster than ever.

And I love it. Why?

Because the workforce is tired.

Tired of 3% raises and 7% inflation.
Tired of being treated unfairly.
Tired of long commutes.
Tired of bad bosses.

So they are saying no to the old normal.

And learning to make money on their own...

Coaching.
Teaching.
Helping.
Selling.

They're moving out of high-cost cities and into lower-cost, tax-friendly areas.

Realizing they can support themselves.

Armed with nothing but an internet connection and their own knowledge.

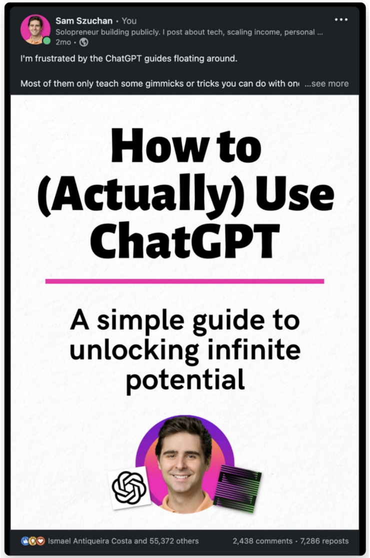
And they won't be going back...

Because they don't want dream jobs.

They want dream lives.

✌️

You and 50,314 others · 1,510 comments · 1,573 reposts



Sam Szuchan · You
Solopreneur building publicly. I post about tech, scaling income, personal ...
2mo ·

I'm frustrated by the ChatGPT guides floating around.

Most of them only teach some gimmicks or tricks you can do with on ...see more

Ismael Antiqueira Costa and 55,372 others · 2,438 comments · 7,286 reposts

Elon Musk
@elonmusk

Next I'm buying Coca-Cola to put the cocaine back in

6:26 AM · Apr 28, 2022

Problem

Viral sequences

- Posts which accumulate very high views
- Definition: Absolute vs relative
- Social media, youtube, etc.

Problem - Given a social media post, predict if it will go viral in the next K days.

Significance

- Correlation between viral and spam
- Improve user experience

Formal setup

Input - for every sample (post)

- Author information
- Post history (timestamps of views, like, share, comment)
  - $H = \{(t_i, z_i) \mid t_i < t\}$

Prediction problem:

- If the sample will be ‘viral’ in the next K days
- Samples identified positive may be queued for manual review.

Discussion ?

A simple featurizer

Let the current time = t

Construct relative time differences

- Given a sequence S0 = {t0, t1, …, tn} of views
    - transform them to S1 = {t - t_0, t - t_1, …, t- t_n}

Discretization of time differences

- Transform S1 as S2 = {b_0, b_1, …, bn}
    - Each b_i denotes a time bin (say, hourly)

Fixed length feature

- {b_0: c_0, b_1: c_1, …, b_n: c_n}

A simple featurizer

Components

- Count features for Views, likes, shares separately.
- Features for the author
    - Demographic
    - Network features
        - how many connections
        - how frequent viral posts
        - How frequent spam posts

Assumptions

- Each view / share / like is independent
- Only statistics over each time window are important.

Model learning

Dataset construction

- Randomly select samples
- For each sample
  - Label each sample as +/-
  - If sample is +
    - Let the virality timestamp = t
    - Choose a timestamp t_f in [t - K days, t]
    - Construct the count features for each sample upto time t_f
  - If sample is -
    - Let current time = t
    - Choose any timestamp t_f < t - K days
    - Construct count features for each sample upto t_f

Models: LR / SVM / Trees, etc.

Model Evaluation

Metrics of interest

- Recall@k
- Precision@k

Results

- An XGBoost prediction model was used
- At recall level of 50%, the precision was 80%

Reasonable results: Can they be improved ?

Limitations

Viral actions -> views

- The proposed model doesn’t make use of it

Features correlation

- Counts in a time bin may have correlation with future ones.

How to include these into the prediction model ?

Poisson Process

Definition: Counting Process N(t)

- Counts #events upto time t
- N(t) >=0, Monotone

Homogeneous Poisson process

- Special case of counting process
- For $\lambda > 0$, $N(t) \sim \text{Poisson}(\lambda t)$
    - Eg. lambda = 1, t = 1
    - N(1) ~ Poisson(1)
    - N(2) ~ Poisson(1*2)
-
- $P(N(t) = n) = \exp(-\lambda t)\ (\lambda t)^n / n!$
- $N(t_2) - N(t_1) \sim \text{Poisson}(\lambda(t_2 - t_1))$

Properties

- $E[N(t)] = \lambda t$
- Has independent increments
- Popular for modeling queues

Poisson Process

Limitations of homogeneous Poisson process

- Past events do not influence future events
- Constant rate λ may be limiting the applications.

Non-Homogeneous Poisson Process

- Assume that λ is a function of time, $\lambda(t)$
- $N(t) \sim \text{Poisson}(\int_0^t \lambda(t)\, dt)$

How to model $\lambda(t)$?

- Past events should influence future events
- Influence of an past event should reduce with time

Hawkes Process

Influence model

- $\lambda(t) = \mu + \Sigma_{ti < t}\, \varphi(t - t_i)$
- μ: modeling the event occurring on its own
- $\varphi(t - t_i)$
  - model the event happening at t, because of a past event at t_i.
  - φ should reduce as $(t - t_i)$ increases
  - Common choice $\varphi(t - t_i) = \alpha \exp(-(t - t_i))$
- $\lambda(t) = \mu + \alpha\, \Sigma_{ti < t} \exp(-(t - t_i))$
  - Parameters : $\mu>0$, $0<\alpha<1$

Hawkes Process for virality

Mapping to virality problem

- Separate counts $N(t)$ for number of views, likes, shares, comments, resp.
  - $N(t) = [N_{view}(t), N_{like}(t), N_{share}(t), N_{com}(t)]$
- Should capture interactions
  - Like by an user A causes view by an user B
  - Share by an user A causes share by an user B
- Need separate $\lambda(t)$ for each type
  - $\lambda(t) = [\lambda_{view}(t), \lambda_{like}(t), \lambda_{share}(t), \lambda_{com}(t)]$

Hawkes for multiple variates:

- α assumes all previous events have same influence
- Relax this: $\alpha_{i, j}$ should depend on the source and target event types
-
- $\lambda_{view}(t) = \mu + \Sigma_{ti < t}\, \alpha_{z(i), view} \exp(-(t - t_i))$

Hawkes for prediction

Given a sample (post),

- Compute the best parameters μ, α specific to the sample
  - Maximum likelihood estimation
- Now, μ, α may be thought of as a feature representation for the sample
- Combined features
  - Binned counts
  - μ, α
- Train a binary classification model
  - Similar to the previous attempt

Entity Resolution

Problem setup

Real estate transactions

- Agent A, date, value, company, license, contact info, etc.
- Noisy data
  - Lack of an unique identifier for the agent
  - License, contact info typically shared across peers
  - Names misspelled, abbreviated, sub with nicknames, etc.

Problem

- Need a way to attribute a transaction to an agent
- Similar problem: normalize the company names
- Total number of transactions ~ O(Millions)

Solution approaches

Requirements

- Name comparisons: Name 1 == name 2 ?
- Comparison of tuples:
  - (name1, license1, contact1, company1, ….)
  - (name2, license2, contact2, company2, ….)
- Not scalable to compare all tuple pairs.

Name comparison

- Word embeddings ???
- Simpler distance measures
  - levenshtein distance
  - Extension for word level
- Hashing based
  - LSH: Locality sensitive hashing

A scalable approach

Goal

- To generate clusters among the tuples
- Issue: can’t compare all pairs

Two stages

- Generate macro clusters
  - Lower precision: a cluster may have multiple entities
  - High recall: All records of an entity to fall within the same cluster
- Refine macro clusters
  - Induce a graph across the tuples as nodes
  - Compare all pairs to arrive at edge weights
  - Identify strongly connected components within.

Generating Macro clusters

Attribute driven

- ‘Group by’ the attributes, say license
- Aggregate the names within a group (licence id-> list names -> cluster the names)
  - Cluster the names
  - Use Levenshtein distance
- Scalable for very large data in spark
- Avoid false positives
  - Qualify each attribute value for clustering
  - Intuition: An attribute mapping to many (say, 100) different names is more likely to generate false positive clusters
  - Entropy based qualification.

Name driven: LSH

- Core idea: higher the similarity of a pair, higher the likelihood that they are hashed to the same bucket.
- A hash bucket represents a macro cluster.
- Possible to derive hash buckets without all pairs comparisons.

Refining Macro clusters

Graph construction

- Nodes: each tuples within a macro cluster
    - Tuple: (name, license, contact, company, …)
- Edges:
    - Non-weighted - have an edge if two tuples share any of the attribute
    - Weighted - fraction of attributes shared
    - Learnt model: train a model to predict the similarity of the tuples
        - Requires labelled dataset
        - Model output required to represent P[tuple1 == tuple2]

Graph clustering

Connected components

- Set a threshold W for weights
- Drop all edges whose weight < W
- Identify all connected components (islands) as an entity

Spectral clustering

- An approach for identifying communities
- Given a similarity matrix A
    - Construct Laplacian $L = D - A$
    - D: Diagonal matrix, $D_{ii} = \sum_j A_{ij}$
    - If unweighted A, D_{ii} represents the degree of node i
    - K-dimensional Embedding for nodes: First K Eigen vectors of L, ordered by eigen value
    - Use k-means on the node embeddings

Questions?

Thank you!