Topic Modelling Algorithms in Information Retrieval and Filtering

Final Year Project
Final Report

Jack Power
20080169
Supervisor: Rob O'Connor
BSc (Hons) in Applied Computing (IoT)

Introduction

The purpose of this project is to investigate and compare the class of machine learning algorithms which use *topic modelling* to extract latent features from data. Topic modelling algorithms can be applied to a wide array of machine learning tasks, from classic natural language processing (NLP) problems such as sentence similarity, to predicting user ratings for recommender systems. Using the Amazon SageMaker service, an experiment was constructed to train and test a variety of machine learning models, both supplied by SageMaker and included from external libraries. The particular task demonstrated is movie recommendation, using the MovieLens 100k dataset to train and make predictions of how users will rate unseen movies. The prediction error is reported by each model, and these validation metrics are compared and the models ranked accordingly. Comparison is made between the experimental results obtained and the example benchmarks supplied by Amazon.

1. Literature Review

Topic modelling algorithms first found use in NLP applications, where *topics* have the conventional meaning of “what a document is to do with”. Given a body of documents, a topic model is constructed by analysing words which commonly occur together and deriving some common topic. Each document can then be considered to be a weighted sum of topics. This has immediate application in the contexts of information retrieval and filtering, where a query implies a topic and all documents related to the topic should be returned to the user.

A fundamental concept underpinning semantic analysis is the distributional hypothesis, the assumption that words with similar meanings will occur with similar distributions. This is more eloquently expressed by John Firth (1957), ‘*You shall know a word by the company it keeps.*’

The first application of these techniques to information retrieval was in the seminal paper of Latent Semantic Analysis (LSA) (Deerwester et al, 1990), initially called Latent Semantic Indexing (LSI) in information retrieval circles. In the paper, several problems facing preexisting methods are outlined. Basic term overlap schemes, where terms in a query are matched against terms in the documents, are severely impacted by the phenomena of *synonymy* and *polysemy*. Synonymy refers to when multiple words convey the same meaning. Polysemy refers to when a single term has many possible meanings, depending on the context. For instance, the word “chip” has different meanings in the contexts of computing versus cooking.

These idiosyncrasies of language create issues when a human is tasked with describing the documents they are looking for. They may by chance use every word applicable to a topic except for the one which appears in the document. They may also experience frustration when in searching for literature on chip design they receive cooking recipes instead. This unreliability of term overlap culminates in the following realisation. Any particular document can be considered to be only a small selection of all the possible discourse on its associated topic. Similarly, the query string is only a small selection of words which may be associated with the implied topic. Thus, extracting index terms from either is inherently fallible, there always exists the possibility of omission on either end. What the user truly desires is to retrieve all documents associated with the topic that this query implies, the “latent semantics”.

This approach largely eliminates the problems mentioned earlier. Given a well-developed body of existing documents, which are now represented in terms of their topics, the model can make associations based not on term overlap, but the topic implied by the words in the query.

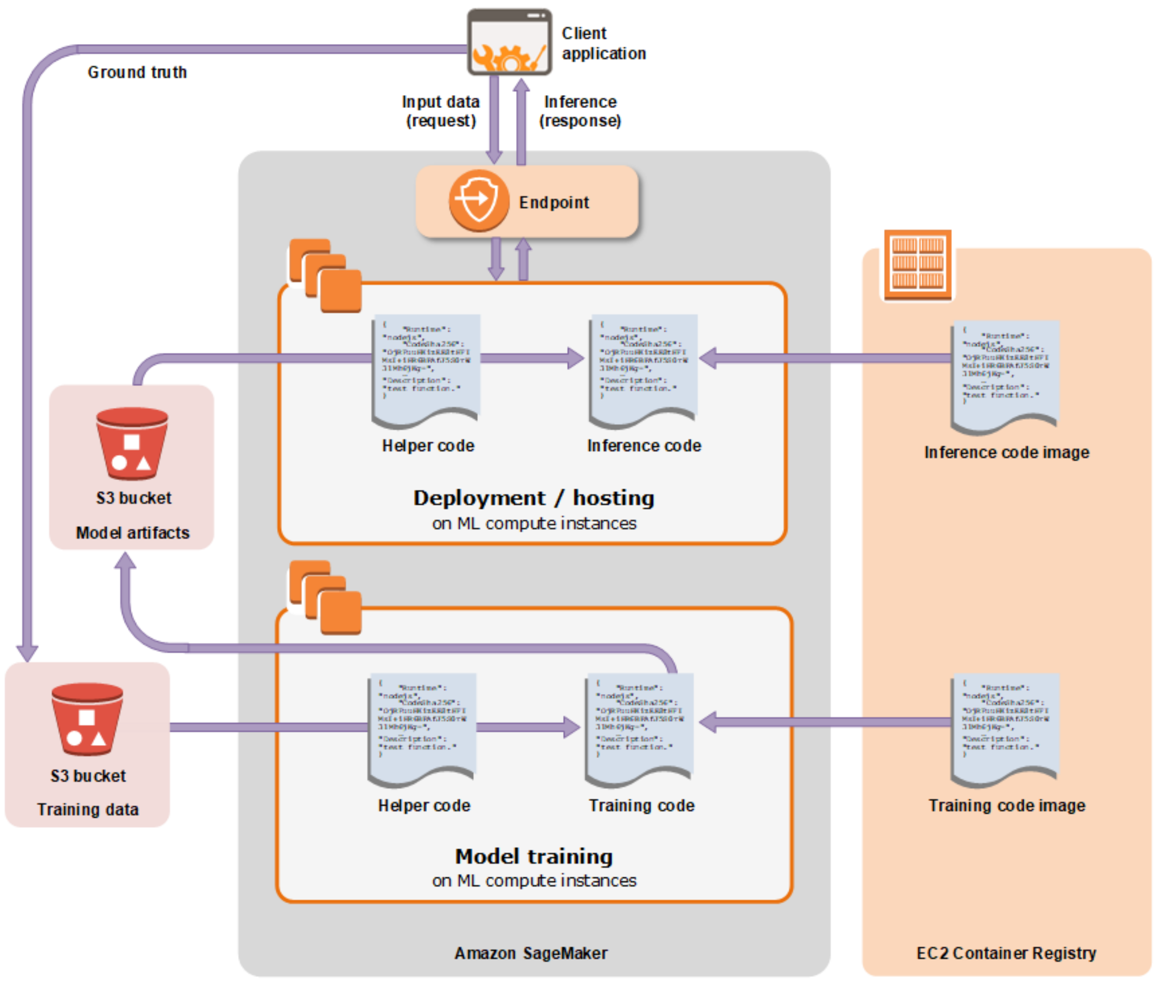
In the introduction of LSA, the technique compared very favourably to contemporary term overlap methods. This novel approach of modelling topics had inherent advantages which led to much interest and further development. A more recent example is the word2vec algorithm (Mikolov et al, 2013), a neural network model which aims to preserve linear regularities among words. The vector representations of words generated by word2vec have captured subtle semantic relationships to such a high degree that performing algebraic operations on the vectors yields intelligible results. For example, the vector representation of the word “King” is to that of “Man” in the same way that the vector of “Queen” is to that of “Woman”.

The techniques used to learn semantic relationships for word similarity tasks can be extended to many other applications where items can have some notion of similarity. A commonplace example of where such topic modelling is useful is in recommender systems, such as there are on many media streaming services. Amazon SageMaker provides the object2vec algorithm (AWS, 2021a), a generalisation of word2vec which is capable of embedding generic objects, preserving semantic similarity in the same fashion as with text. In the case of recommender systems, similarity corresponds to how highly a user rates a given item. Topic modelling algorithms are capable of learning the hidden features which dictate how certain users rate certain items by analysing previous ratings. This technique of predicting the rating of a single user based on the previous ratings of others is known as collaborative filtering. A high-profile example of such a recommendation task is the Netflix Prize (2009). From 2006 to 2009 Netflix held an open competition for teams to submit algorithms which would score better than their own given a dataset of movie ratings. Each rating contained only a numerical user and movie ID, a rating from one to five and the date of the rating. The portion of the dataset used for training consisted of over 100 million ratings from 480,000 users for 17,000 movies. The metric used to score predictions is RMSE (root-mean-square error), that is the absolute deviation of the prediction from the true rating which was unknown to the model. The algorithm which ultimately won was BellKor’s Pragmatic Chaos which achieved an RMSE of 0.8567, a 10.06% improvement over Netflix’s own Cinematch (RMSE 0.9525).

For this experiment the same metric will be used as it is exceedingly common and provides easy comparison between different algorithms. The dataset to be used is the MovieLens 100k dataset, 100,000 ratings from 943 users for 1682 movies (GroupLens, 2021). This is a well-known dataset, and should prove much more manageable than one as colossal as the Netflix dataset.

In order to construct an experiment of sufficient rigor and repeatability, a scientific testbed is used in the form of Amazon SageMaker (AWS, 2021b). SageMaker allows machine learning models to be trained and tested in a consistent environment with powerful tools available to organise and manage experiments. Many examples are provided in the form of Jupyter notebooks which can be run locally or on the web-based machine learning IDE SageMaker Studio (AWS, 2021c), which provides additional features and libraries for interacting with SageMaker.

Shown is a diagram illustrating the training and deployment of a model using SageMaker (AWS, 2021d).



The training code image is first loaded into an ML compute instance and trained on the data provided via an S3 bucket. There are a range of high-performance algorithms built into SageMaker which greatly simplifies their use. It is also possible to provide custom algorithms which may be dependant on any of the machine learning frameworks which SageMaker supports (TensorFlow, PyTorch, Apache Spark, etc.).

The generated model artifacts are passed into a destination bucket which constitutes the trained model. The inference code image is then loaded which exposes the model for querying via an endpoint. Shown at the top is a client application requesting inference (a prediction) based on some unseen data.

2. Methodology

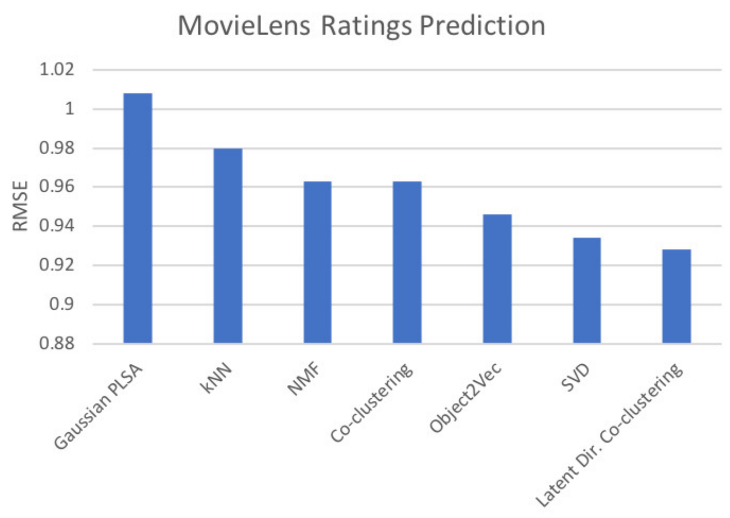
In investigating the SageMaker example notebooks covering concepts relevant to the project, a variety of machine learning problems were comprehensively demonstrated. As discussed, the classic application of many topic modelling algorithms is to common NLP tasks, such as sentence similarity or information retrieval. However, the mathematical techniques and models which drive these algorithms can often be generalised to broader applications.

A prominent example of this on SageMaker is the two built-in algorithms BlazingText (AWS, 2021e), a highly-optimised implementation of word2vec, and its generic offshoot object2vec. The SageMaker example for object2vec (AWS, 2019) demonstrates its application to a recommendation problem, namely making movie recommendations to users based on previous ratings. Many aspects of the example make it particularly attractive, movie recommendations are a common feature of many well-known platforms and services. As opposed to some of the more abstract use cases for these algorithms, it is a problem which is easy to conceptualise and communicate.

Furthermore, the dataset used is exceedingly simple. The MovieLens 100k dataset contains pairs of user IDs and movie IDs, followed by a corresponding score from one to five of how that user rates that particular movie. Based on this data, the problem can now be reduced to a regression task where, given a user and movie pair for which the rating is unknown, the model must attempt to make a prediction of their rating.

The example notebook demonstrates the steps required to train the object2vec algorithm on the MovieLens dataset, then deploying the trained model to an endpoint and evaluating the performance. In order to provide context for the performance of the algorithm, the notebook gathers corresponding metrics for a variety of other rating prediction algorithms.

Shown is the RMSE prediction error for object2vec and other recommender algorithms.



(AWS, 2019)

The metrics shown are benchmarks declared by the recommender libraries which supply the algorithms. One of these libraries is the Surprise Python scikit (Hug, 2020) which provides a range of algorithms specialised for rating prediction tasks.

In order to create a more complete experimental context, it was decided to assemble these algorithms on SageMaker and to train and evaluate them alongside object2vec. This would allow for a greater degree of control and fuller understanding of the models and their associated metrics. The resulting framework also provides a template for other models to be included later. The well-developed toolset of Amazon SageMaker makes the management of machine learning experiments consistent and extensible, as well as being inherently self-documenting.

A particularly useful feature is SageMaker Search (AWS, 2021f), providing the ability to query and present key information about models trained using SageMaker, such as training and validation metrics, data sources and hyperparameters used. Each model trained has its attributes automatically published to CloudWatch for monitoring and analysis, including user-defined tags to organise and classify experiments.

3. Implementation and Experimentation

Modelled after the Amazon example notebooks, a notebook was created using SageMaker Studio that would perform the necessary data preprocessing and preparation before training each model on the dataset and comparing the resulting metrics.

3.1. Dataset

The MovieLens 100k (ml-100k) dataset is first read in from a remote url and stored locally. It consists of 100,000 ratings from 943 users on 1682 movies. The data is in tab-separated CSV format with fields user ID, movie ID and rating. Also present is a Unix timestamp for the rating, some models can use this information to further improve their predictions but it is not included in this experiment.

Shown are the first five entries in the dataset (entries are randomised).

userID	movieID	rating	timestamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596

The dataset provides some extra information about both the users and movies, but all that is needed are the user and movie ID pairings, along with their associated rating.

The ml-100k dataset comes with premade training and testing splits of the data for a variety of use cases.

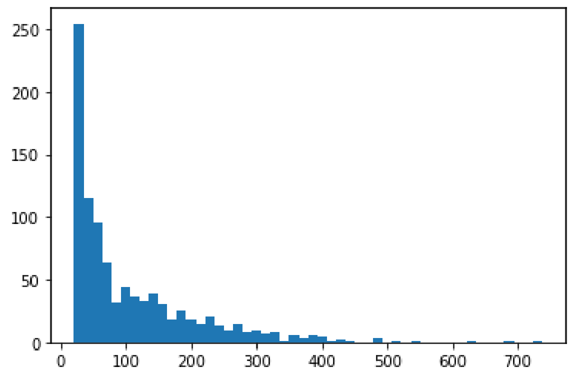
For the purposes of this experiment the breakdown is as follows:

- `ua.base` and `ua.test` are used for training and validation respectively, between them they hold all 100,000 ratings
- `ua.test` contains 9430 of the total 100,000 ratings, a 9.43% split with exactly 10 ratings per user.
- `ub.test` is used for conducting external testing, it too has 9430 ratings, none of which appear in `ua.test`

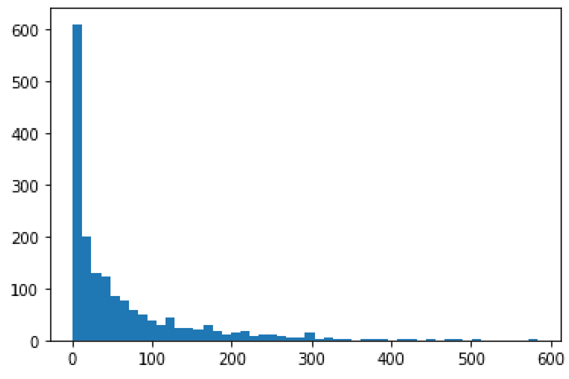
3.2. Exploratory Data Analysis

Some basic exploratory data analysis is carried out to showcase some of the dataset's characteristics and to verify some assumptions about the data.

```
Ratings per user:
    Min: user 926 with 20 ratings
    Max: user 405 with 737 ratings
    Median: 65 ratings
```



```
Ratings per movie:
    Min: movie 1653 with 1 ratings
    Max: movie 50 with 583 ratings
    Median: 27 ratings
```



3.3. Data Preparation

The dataset, now held in memory as a Pandas data frame, must be re-encoded and uploaded to an S3 bucket so that it can be consumed by the algorithms during training. The object2vec algorithm requires JSON Lines input, while all algorithms from the Surprise Python scikit require CSV data. For this reason the dataset is replicated in both formats and each is uploaded to S3.

3.4. Training

Each algorithm is then trained on the preprocessed datasets.

3.4.1. Docker

The algorithms Amazon supplies are actually Docker images which must first be retrieved from the Amazon Elastic Container Registry (ECR) (AWS, 2021g). Docker containers allow a standard interface to supply the necessary training and validation data, hyperparameter values and other inputs to the model which can then be accessed and manipulated internally.

In keeping with this model, each Surprise algorithm to be trained is built into its own Docker image and pushed to the ECR. They can then be used as any other SageMaker algorithm, simply provide the necessary input channels and allow any unique requirements or processing to be satisfied within the container’s code.

Amazon provides an example of building custom Docker containers using the scikit-learn decision tree classifier (AWS, 2020). For the level of complexity required for this experiment, the only modifications needed are to the training and prediction modules, as well as installation of the required libraries when the image is built.

3.4.2. Search

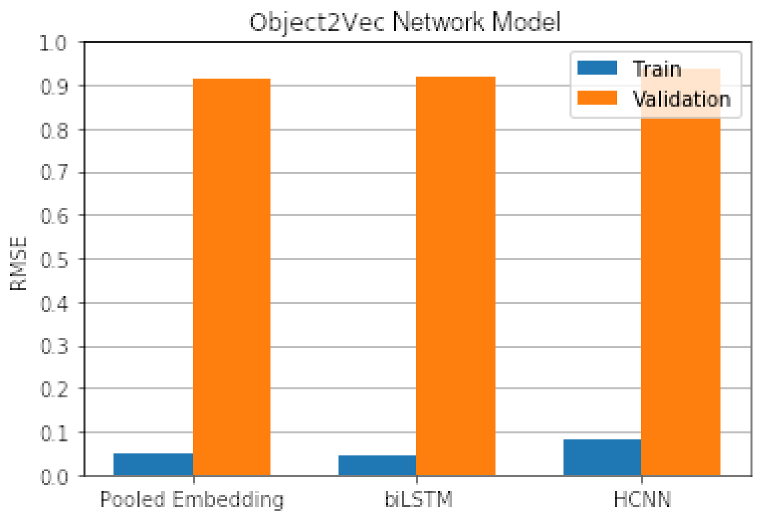
Training a model on SageMaker results in what is known as a `TrainingJob`, which encapsulates the attributes mentioned previously such as hyperparameter values and validation metrics. These values can then be queried using SageMaker Search. A variety of predicates and filters are available to tailor the search and organise the results. Perhaps most useful for the purpose of managing experiments is the ability to tag training jobs, allowing for easy identification and categorisation.

Shown is a search query similar to that in the experiment, in JSON format.

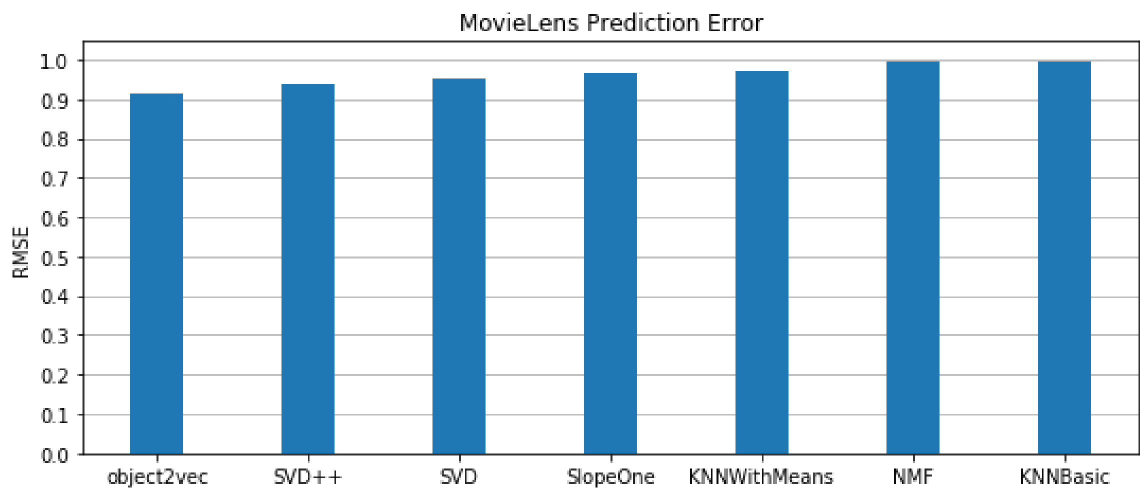
```
{
    "Resource": "TrainingJob",
    "SearchExpression": {
        "Filters": [
            {
                "Name": "Tags.{MyCustomTag}",
                "Operator": "Equals",
                "Value": "{CustomTagValue}"
            }
        ]
    },
    "SortBy": "Metrics.validation:mean_squared_error",
    "SortOrder": "Ascending"
}
```

4. Results and Analysis

The object2vec algorithm offers three network models to choose from. These are HCNN, a hierarchical convolutional neural network, biLSTM, a bidirectional long short-term memory network, and pooled embedding, which averages the embedding of all tokens. These can be selected through a hyperparameter to the algorithm. Shown are the training and validation metrics for each network model.



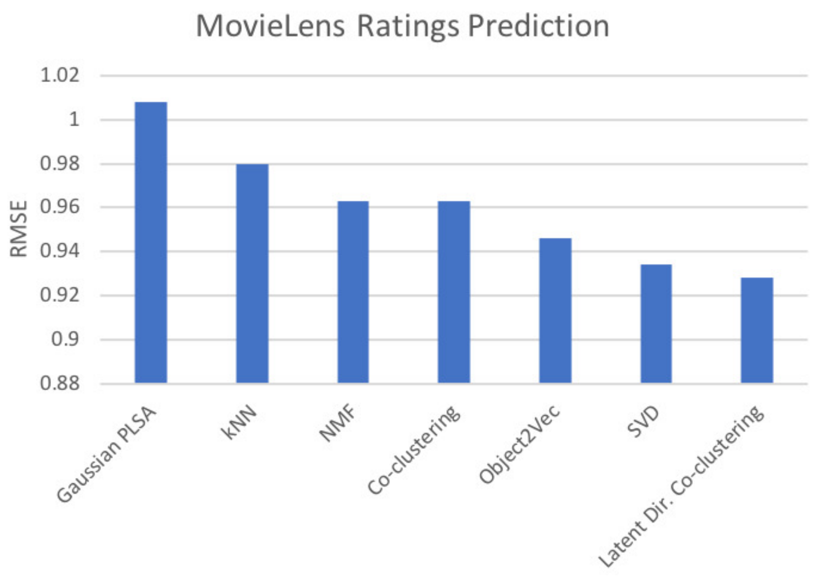
Of these, the best was selected and compared with the validation metrics for each of the Surprise algorithms.



Shown are the RMSE prediction errors for each algorithm, in ascending order.

Algorithm	RMSE
object2vec pooled embedding	0.91340
object2vec biLSTM	0.91738
SVD++	0.93697
object2vec HCNN	0.93775
SVD	0.95020
slope one	0.96488
KNN with means	0.97152
NMF	0.99286
KNN basic	0.99631

Here is the example notebook metrics once again for comparison.



Variation is expected given inherent randomness during training. However the general placement of algorithms present in both graphs is roughly consistent. All but one of the object2vec network models outperformed the other algorithms, with the two SVD techniques close behind. KNN with means, which improves its prediction based on the mean rating of each user, outperformed Non-negative Matrix Factorisation (NMF), while basic KNN did not.

5. Future Work

In order to more fully explore and compare these models, hyperparameter tuning would undoubtedly have great impact on the results. It would also hugely increase the scope of this experiment, not only in the experiment itself but also in the knowledge required to effectively tune each model. Due to time constraints, it was decided to restrict the hyperparameters varied to the three network models of object2vec. However, the existing structure of the experiment should lend itself well to further expansion, in no small part thanks to the robust interfaces offered by the SageMaker ecosystem.

Bibliography

Firth, J. (1957). *Papers in Linguistics 1934-1951*. London: Oxford University Press.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science,* 41(6), pp. 391-407.

Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *The International Conference on Learning Representations*.

AWS, (2021a). *Object2Vec Algorithm - Amazon SageMaker*. [online] Available at: https://docs.aws.amazon.com/sagemaker/latest/dg/object2vec.html [Accessed 2 May 2021].

Netflix Prize, (2009). *Netflix Prize: Review Rules*. [online] Available at: https://netflixprize.com/rules.html [Accessed 2 Mar. 2021].

GroupLens, (2021). *MovieLens 100K Dataset*. [online] Available at: https://grouplens.org/datasets/movielens/100k/ [Accessed 2 Mar. 2021].

AWS, (2021b). *Amazon SageMaker - Machine Learning - Amazon Web Services*. [online] Available at: https://aws.amazon.com/sagemaker/ [Accessed 2 May 2021].

AWS, (2021c). *Amazon SageMaker Studio - First IDE for Machine Learning - Amazon Web Services*. [online] Available at: https://aws.amazon.com/sagemaker/studio/ [Accessed 2 May 2021].

AWS, (2021d). *Train a Model with Amazon SageMaker*. [image] Available at: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html [Accessed 2 May 2021].

AWS, (2021e). *BlazingText Algorithm - Amazon SageMaker*. [online] Available at: https://docs.aws.amazon.com/sagemaker/latest/dg/blazingtext.html [Accessed 2 May 2021].

AWS, (2019). *An Introduction to SageMaker Object2Vec Model for MovieLens Recommendation*. [online] Available at: https://github.com/aws/amazon-sagemaker-examples/blob/master/introduction_to_amazon_algorithms/object2vec_movie_recommendation/object2vec_movie_recommendation.ipynb [Accessed 2 May 2021].

Hug, N. (2020). Surprise - A Python scikit for recommender systems. [online] Surprise. Available at: http://surpriselib.com/ [Accessed 2 May 2021].

AWS, (2021f). *Search Using the Amazon SageMaker Console and API - Amazon SageMaker*. [online] Available at: https://github.com/aws/amazon-sagemaker-examples/blob/master/introduction_to_amazon_algorithms/object2vec_movie_recommendation/object2vec_movie_recommendation.ipynb [Accessed 2 May 2021].

AWS, (2021g). *Amazon ECR | Docker Container Registry | Amazon Web Services*. [online] Available at: https://aws.amazon.com/ecr/ [Accessed 2 May 2021].

AWS, (2020). *Building Your Own Algorithm Container*. [online] Available at: https://github.com/aws/amazon-sagemaker-examples/blob/master/advanced_functionality/scikit_bring_your_own/scikit_bring_your_own.ipynb [Accessed 2 May 2021].