Deep Artificial Intelligence for Fantasy Football Multimodal Media Understanding

Aaron Baughman
IBM
Cary NC, USA
baaron@us.ibm.com

Stephen Hammer
IBM
Atlanta GA, USA
hammers@us.ibm.com

Gray Cannon
IBM
Miami FL, USA
gfcannon@us.ibm.com

Micah Forster
IBM
Austin TX, USA
mforste@us.ibm.com

Jeff Powell
IBM
Atlanta GA, USA
jjpowell@us.ibm.com

Chris Jason
Disney ESPN
Bristol CT, USA
Chris.Jason@disney.com

Daniel Bohm
Disney ESPN
Bristol CT, USA
Daniel.Bohm@disney.com

Sai Gudimetla
IBM
New York NY, USA
sgudime@us.ibm.com

ABSTRACT

Fantasy sports allow fans to manage a team of their favorite athletes. The manager of the team makes decisions of which players to roster and trade based on analysis of sports media and statistics. The sports media industry rabidly produces content to the tune of trillions of bytes of natural language text and multimedia data which is not possible for a human to analyze. Our work discusses the results of a novel machine learning system (in production from 2017) which helps manage a fantasy team. The system analyzes media content: videos, articles, podcasts for a particular player and identifies sentiment, keywords, entities and concepts. The system combines the NLP insights with statistics to come up with 4 player specific measures: boom (player scoring above a certain threshold), bust(player scoring below a certain threshold), play with a hidden injury or play meaningful touches. A fairness postprocessor is applied to remove the bias in coverage for the particular player or team. After that, the system produces a score spread of each player. The keywords, entities and concepts are extracted by trained statistical entity detectors and document2vector models applied to over 100,000 news sources crawled each day. We also use probability density functions to produce a score spread of a players projected points. The resulting visualizations, projections and sentiment analysis were compelling to end users (9.1 million users per month in 2019), as each user spent over 90 seconds (throughout 2019) using the evidence from our novel system.

CCS CONCEPTS

• Computing methodologies → Artificial Intelligence

KEYWORDS

Multimedia, Unstructured Text, Machine Learning, Deep Learning, Sports Analytics, Natural Language Processing

WISDOM '20, held in conjunction with KDD'20, August 24, 2020, San Diego, CA USA© 2020 Copyright held by the owner/author(s). Publication rights licensed to WISDOM'20. See http://sentic.net/wisdom for details.

1 Introduction

Fantasy sports owners and managers have hundreds of critical questions to answer before selecting their team. Who will score the most points this week? What player will be a bust or breakout? Will any players be a sleeper? Do any players have injuries that are going to impact their play? When should a player start to counter my opponent's team? Are there any available trades to upgrade a team? The number of possible moves to make is daunting for both the professional and novice player.

With over 9.1 million unique fantasy football players per month on the ESPN platform alone, the demand for content is insatiable. Every day during the 2018 and 2019 seasons, we sustained 2 billion edge hits and delivered 250 TB of AI content per day. The large volume of users bases the majority of their roster decisions on player rankings and simple statistics. However, unstructured and multimedia information about sports is the largest data component. The volume of natural language, video, and podcast content creates fantasy football content overload that is an epidemic among current team managers.

The overwhelming majority of fantasy sports participants filter content based on personal biases such as reading articles, watching videos, or listening to podcasts about their favorite team or from their preferred outlet. On average, fantasy players consume 3.9 sources to base their decisions [22]. Other users rely on ad hoc tools such as querying statistics databases, excel sheets, or natural language searches [6,22]. The limited amount of information each manager can consume has created tremendous knowledge gaps when making decisions.

Throughout the 2018 and 2019 NFL football seasons, we developed a novel system that reads and comprehends natural language, videos, and podcasts from over 100,000 sources that were deployed to the ESPN Fantasy Football mobile and desktop experiences. The semantic relationships between words and topical understanding through techniques such as doc2vec enabled deep learning classifiers to make decisions about each football player. Team managers and coaches now have insight from unstructured textual, and multimedia data as to which players will be a bust, breakout, play meaningful touches, or play with a hidden injury. Statistical data is combined with our system's comprehension of

unstructured information through a deep machine learning pipeline. A best-fit score distribution from a set of 24 probability density functions (PDF) based on current and historical score projections provide an understandable score spread. To answer the question of "why", we presented the top 10 articles, podcasts, and videos that support or refute our machine learning pipeline's player assessment. This paper depicts the empirical evaluation of our ESPN Fantasy Football Insights with Watson system (FFIW).

2 Abbreviated Related Works

Advanced analytics that use structured data such as historical game statistics are prevalent and widely used by fantasy sports managers. For example, Rotogrinder provides a service that builds starting lineups, player projections, Vegas odds, depth charts, and weekly weather reports. Another tool available for fantasy sports players is called Dailyfantasynerd. The service highlights favorable statistics for players, maintains a lineup optimizer, and displays weather data for each venue. Fantasyfootballanalytics.net exposes aggregated play statistics for analysis along with custom point projections and player risk assessments.

Tools that ingest, consume and utilize a wide range of unstructured data such as text and multimedia have had limited utility for direct team management. On rotowire.com, fantasy sports managers can ask a human expert that has curated both structured and unstructured data for advice. ESPN has an insider paid service to access premium content written by featured columnists as well as roster advisors. SportsQ has a natural language question and answer system to retrieve passages relevant to a question. However, none of the prior work distills millions of articles, videos, and podcasts every hour into AI insights.

2.1 Machine Learning in Fantasy Football

All of the prior work addressing fantasy sports has centered around statistics and structured data. We are not aware of any previous works that have analyzed multimedia and text information for the basis of computational guided fantasy sports play. For example, Landers and Duperrouzel present several machine learning approaches for predicting points scored by players as well as strategies to optimize a team. Other works use statistical predictors from sports play to optimize teams [25]. Following the general body of work within fantasy sports, Hermann et al. show how regression, naïve Bayes, and decision trees can be used to predict fantasy basketball performance. Other works analyze predicting or projecting players' values within fantasy sports [12,18,24]. Seal addresses any doubt that machine learning with statistics can improve fantasy play. We further the state of the art and demonstrate how multimedia and natural language processing can provide quantitative and qualitative benefits to fantasy football experiences.

2.2 Deep Learning for Natural Language Processing

Large-scale text classification has been inspired by the growth of natural language text over social media and news outlets. Work by Glorot showed that stacked denoising auto-encoders performed text sentiment classification better than Support Vector Machines (SVM), Structural Correspondence Learning (SCL), Multi-label Consensus Training (MCT), and Spectral Feature Alignment [28]. Other deep learning works used convolutional neural networks and transfer learning by sharing network levels for auxiliary tasks to provide Semantic Role Labeling (SRL) [17].

Seminal work culminated by Tomas Mikolov in 2013 outlined the beginnings of the application of deep learning to natural language processing with doc2vec. Billions of words can be added to a computing system's vocabulary, which progresses the traditional n-gram language model [27]. Additional work shows that distributed word vector representations improve text classification over Bag of Words (BoW) and Support Vector Machines (SVM) [12].

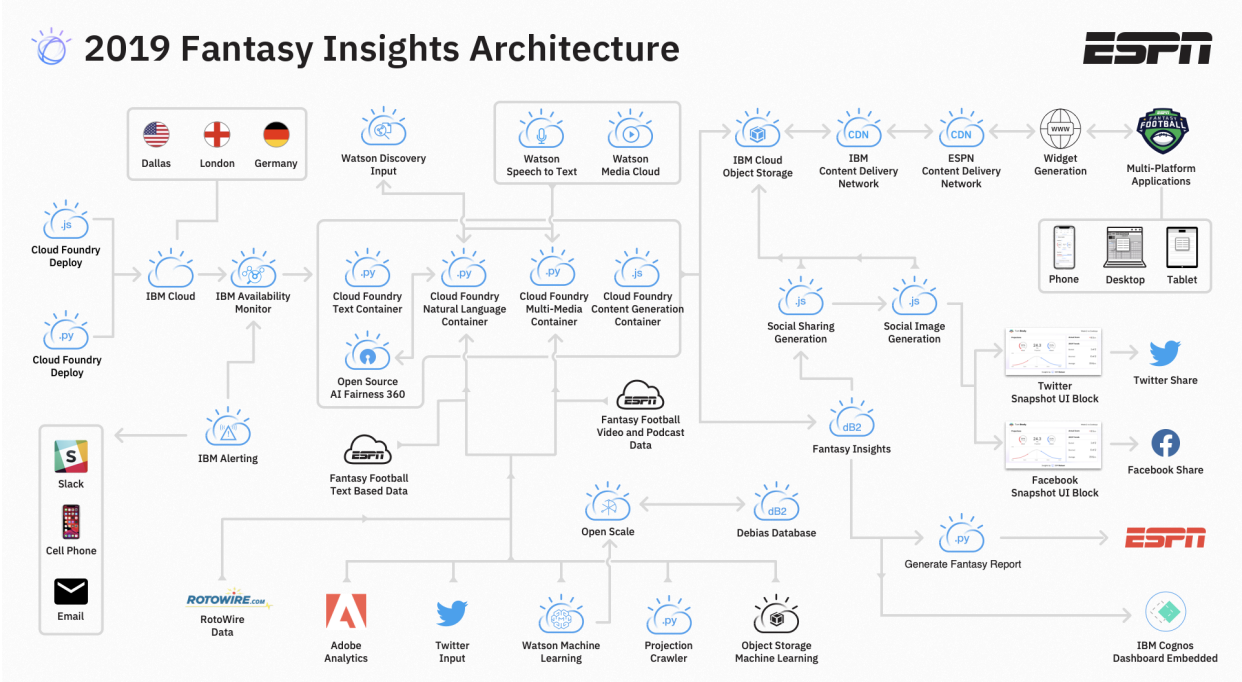
Works began to use doc2vec approaches to expand short text for improvements in text classification [13]. Other works enriched short text using tf-idf measures before using a doc2vec approach for text classification [5]. Text embeddings were combined with multimedia convolutions within multitask learning to improve model performance [9]. Word2vec approaches are used within language translation to emphasize the use of unlabeled data between monolingual data [23]. Unlike the previous works, we extend doc2vec by summarizing thousands of documents into entities, concepts, and keywords before creating average word embeddings.

2.3 Deep Learning Fairness

A lot of work and research has been focused around ethical and fair computing. In particular, AI Fairness 360 is an open source Python library that has dozens of bias identification and mitigation techniques [15]. Within the prior work, bias mitigation can be implemented at any stage of a machine learning model such as pre, post or inline [15]. In Reubenn Binns's work, fairness is defined from multiple perspectives that influences computing. Other works such as Narayana et al. mathematically define fairness. With a focus on deep learning, M. Dhu et al. provide a survey around fair neural networks. This is particularly important so that we can provide fair player states independent of team popularity.

2.4 Text Based Sentiment

Recently, in the field of Natural Language Processing (NLP), there has been an emergence of several transfer learning methods [2,4,13]. In transfer learning, data can be leveraged from different domains or tasks to be adapted for specific domains. For example, in sequential transfer learning, a model is pretrained on a large unlabeled text corpus and then adapted to a supervised target task using labeled data. Vaswani et al. showed that the transformer architecture based on attention mechanisms has a significant improvement on a variety of NLP tasks. The work has been further improved by applying bidirectional training in Bidirectional Encoder Representations from Transformers (BERT). BERT itself



has been improved upon with a General Pretrained Transformer Model on training time and accuracy [27].

The transformer model aims to learn word embeddings that incorporate both word-level characteristics and contextual semantics. A well-trained transformer model on a large corpus can be adapted to a variety of tasks by modifying the architecture or adapting weights by adding linear layers on top of a pretrained model. Our system uses a pretrained language model that has an adapted transformer and classification layers for the fantasy football domain. The assigned sentiment label to each piece of evidence helps users to understand the media buzz around each player.

3 Overall Architecture

The architecture of the system runs on a hybrid cloud that consists of the IBM Public Cloud, ESPN Cloud and Akamai as shown in Figure 1. On the IBM Cloud, five core Python applications support

AI insights gathering for fantasy football. Each of the applications is deployed as a container-based Cloud Foundry application. Manifest files describe each application that can be deployed with several cloud foundry directives. Each of the applications was targeted and deployed to 3 different sites to maintain continuous availability of services. Monitors were placed on each application during production work loads. RESTful endpoints were called by availability monitors that ran within different geographies, look for HTTP 200 codes. If an error code was retrieved, the monitors send messages to IBM Alerting services through webhooks. Policies within IBM Alerting determine when, who, and how to contact operational support. With millions of users each day, the monitoring provided an alert system that resulted in no outages during the 2019 season.

The AI work was written in Python and distributed across several IBM Cloud and external services. The following list of applications support the machine learning pipeline depicted in Figure 2.

- Natural Language Container (NLC) (Python) = The NLC application wraps the machine learning pipeline around a scheduler. Several Swagger API are exposed to accept real-time player run requests.
- Text Container (Python) = Crawls specific sources to find text articles about each player, enrolls the source into Watson Discovery (WD) and saves state to a DB2 database.
- Multi-Media Container (Python) = Searches sites for videos and podcasts about each player. The media is transcribed with speech to text services and saved into WD for consumption by the Natural Language Container.
- Projection Crawler (Python) = The crawler finds any players that have changes in state such as score projections and injury status. Requests are sent to NLC to process these players.
- Sentiment App (Python) = Runs in Red Hat OpenShift to determine the player buzz surrounding articles, videos and podcasts.
- Content Generation Container (Node.js) = Receives messages through Swagger API's to pull finished AI insights about each player from DB2 for publication to Cloud Object Storage.
- Social Sharing Generation (Node.js) = The application generates HTML index files for each player link to images stored on the Content Delivery Network (CDN). The files can be shared on Facebook and Twitter.

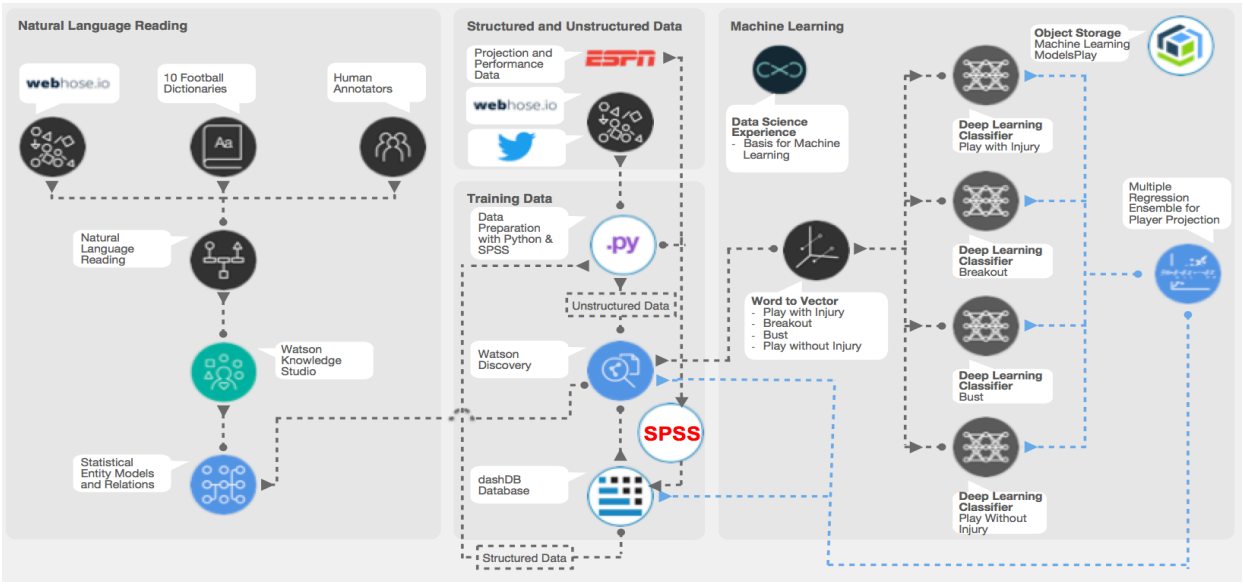


Figure 2: Machine Learning Architecture

- Social Image Generation (Node.js) = Creates snapshots of player images that are uploaded to the CDN and linked by sharable HTML files.

The core machine learning pipeline is written with the Keras and TensorFlow libraries. Each model is built and evaluated offline. The training of each model within the machine learning pipeline takes 6 days of continuous computing. The deep learning offline training uses a total of 93,136 exemplars with 3-fold cross validation over 3 seasons. The score projection phase uses 15,969 exemplars to train an ensemble of models based on player position that predict score projections. All of the training data was derived from a third-party provider, Webhose, that ran historical queries against an archive of the Internet. In total, they provided 100 GB of historical text information while ESPN enabled access to historical player statistics. Tools such as SPSS and Python libraries such as pandas and NumPy were used to analyze offline data.

To ensure that player boom and bust models were fair across all teams irrespective of popularity, the AI Fairness 360 library was implemented within as a pipeline post processor. The Calibrated Equal Odds Postprocessing Python class was trained and tested with 1,550 exemplars. The offline machine learning training used an equal distribution of favorable and unfavorable labels based on team membership.

During runtime, player queries were federated to IBM Watson Discovery that is a machine reading engine for over 100,000 sources, Twitter, ESPN statistics and Rotowire for injury data. Features are extracted from the responses of each query and input into the appropriate phases of the machine learning pipeline as discussed in Section 4. The entire system was multithreaded with 20 threads and joined at each dependent machine learning phase to increase live player throughput. When the initial machine learning pipeline is loaded, the process downloads all models to disk from Cloud Object Storage (COS).

The data produced by the system is stored within a highly available DB2 data warehouse. Trained models and artifacts are placed within COS while system configurations are uploaded into a Cloudant database. The Content Generation Container pulls data from DB2 and generates JSON files that are also stored within COS, which acts as the origin for the two Content Delivery Networks (CDN). To handle the billions of requests for our AI insights, an ESPN CDN fronted the IBM Cloud CDN. If the time to live (TTL) for data expired or if data was updated, client requests populated down to the COS origin to pull and cache the data within both CDNs for follow-on requests. Only 0.06% of traffic reached our origin server.

4 Machine Learning Pipeline

The machine learning pipeline is comprised of natural language understanding of media sources, deep learning networks, debias algorithms and player performance spreads. The deep learning models produce player states such as performance boom, play bust), play with a hidden injury or play meaningful touches. The debiasing algorithms include a fairness post processor to account for bias in the media coverage surrounding a player or team. We then produce score spreads for a player projection by finding the best fit probability density function. Through sampling over the PDF, we approximate a mean player performance. The implementation of the machine learning pipeline is supported by five applications, dozens of models, several data sources, and many data science environments. Figure 2 depicts the overall data flow within our system. Natural Language Understanding of Sports data First, the system had to be taught to read fantasy football content. A novel language model was designed with custom entities and relationships to fit the unique language people use to describe players and teams in the fantasy football domain. Next, an annotation tool called Watson Knowledge Studio was used by 3 human annotators to label text within articles as any combination

of 13 entity types such as player, team, performance, etc. With this data, a statistical entity detector was trained and deployed to our system called Watson Discovery (WD) that continually ingests sources from over 100,000 sources. Podcasts and videos are transcribed and ingested into WD. The WD system is able to discover fantasy football entities, keywords, and concepts from the continually updating corpora based on our trained statistical entity model.

Next, the system used a document to vector model to understand the natural text from a query. A very specific query was initially issued to WD such as "Tom Brady and Patriots and NFL and Football." If a query did not return at least an experimentally determined 50 documents, the query was broadened until it only had "Tom Brady and NFL." From the query result, a list of entities, keywords and concepts for each document was converted to numerical feature vectors. Each of the feature vector groups was averaged together to represent a semantic summarization. All of the feature vector groups from each document were averaged across all documents. The 3 keyword, concept and entity averaged feature vectors, along with player biographic data were input into the deep learning portion of the pipeline.

Deep learning

The deep learning pipeline phase had 4 models that were over 98 layers deep. The models were classifiers for each player to determine the probability of a boom, bust, play with a hidden injury or play meaningful minutes. The probability scores provide a confidence level of player states so that team owners can decide their own risk tolerance.

Fairness

At the end of the of the deep learning phase, a fair post processor ensured equal equity across players on different teams. For example, players on popular teams such the Rams would unfairly have more players predicted to boom and less to bust based on the conversation of the crowd. As a result, each of the players were split into privileged and unprivileged groups. The output of boom and bust probabilities was slighted changed based on team membership.

Spread

Finally, the outputs of the deep learning layers, along with structured ESPN data were input into an ensemble of multiple regression models. This merging of natural language evidence with traditional statistics produced a score projection for every player. On average, the combination of structured and unstructured data produced a better RMSE than each independently. Finally, 24 PDF's were fit to the score projection and historical score trends to produce a player score distribution. While defining and refining these techniques, our team conducted data exploration in Jupyter notebooks and SPSS. Through experimentation, we selected model hyperparameters and algorithms.

4.1 Fantasy Football Training Data

Throughout the project, we used historical news articles, blogs, etc. associated with players from the fantasy football seasons 2015, 2016, and 2017. A third party named Webhose provided the large-scale content. In total, over 100 GB of data was ingested into WD using our custom entity model. We correlated the article date with structured player data from ESPN to generate labeled data. The ESPN player data contained several statistics that included week result, projection, actual, percentage owned, etc.

A week span date from Tuesday to the following Monday was associated with each player state so that a time ranged query could be run to retrieve relevant news articles. Equations 1-3 depict the determination of a boom label. If the actual score of a player was greater than 1 standard deviation above the projection for a player p at a specific position, the weighted average of the differences between the actual and projected score by the percentage owned, $perowned_p{}^{0.1}$, is used to determine a boom standard deviation. However, the player must be owned by at least 10% in all leagues.

$$\mu_{bo} = \frac{1}{N}\sum_{p=0}^{N}\frac{(actual_p - projected_p)}{perowned_p{}^{0.1}}; actual_p > (projected_p + \sigma_p) \quad (1)$$

$$\sigma_{bo}^2 = \frac{1}{N}\sum_{p=0}^{N}\left(\frac{(actual_p - projected_p)}{perowned_p{}^{0.1}} - \mu_{bo}\right)^2; actual_p > (projected_p + \sigma_p) \quad (2)$$

The standard deviation for boom, σ_{bo}, is determined by taking the square root of the boom variance, σ_{bo}^2.

The label boom is applied to the player if their actual score, x, is greater than 1 boom standard deviation above the boom mean for the player.

$$boom(x) = \begin{Bmatrix} 1{:}x \geq \mu_{bo} + \sigma_{bo} \\ 0{:}x < \mu_{bo} + \sigma_{bo} \end{Bmatrix} \quad (3)$$

The bust label is calculated by equations 4-6. The average bust score, μ_{bu}, is determined by weighting the difference between score actuals and projection by the same player's projection. However, only actuals that are 1 standard deviation, σ_p, below the projected scores are used within the sample set.

$$\mu_{bu} = \frac{1}{N}\sum_{p=0}^{N}\frac{(actual_p - projected_p)}{projection_p} * \sqrt{projected_p}; actual_p < (projected_p - \sigma_p) \quad (4)$$

$$\sigma_{bu}^2 = \frac{1}{N}\sum_{p=0}^{N}\left(\left(\frac{(actual_p - projected_p)}{projection_p} * \sqrt{projected_p}\right) - \mu_{bu}\right); actual_p < (projected_p - \sigma_p) \quad (5)$$

The square root of the bust variance, σ_{bu}^2, provides the standard deviation threshold to label a player with score x.

$$bust(x) = \begin{Bmatrix} 1{:}x \geq \mu_{bu} + \sigma_{bo} \\ 0{:}x < \mu_{bu} + \sigma_{bo} \end{Bmatrix} \quad (6)$$

The play with injury label was generated only for players that scored greater than 15% of their projected points and they were on the Rotowire injury report as questionable or probable. The play meaningful minutes label was created when a player scored greater than 15% of their projected points and was probable or not on the Rotowire injury report. Each of the four labels was generated for every week of every player within the fantasy football 2015 and 2016 seasons with a subset from the 2017 season.

4.2 Statistical Entity Detection

The system had to learn how to read fantasy football documents, blogs, and news articles and listen to videos and podcasts. In order to read text and transcripts for comprehension, an ontology of 13

entity types were defined that covered player-centric understanding. The entities include body part, coach, fans, gear, injury, location, player, player status, treatment, positive tone, negative tone, team and performance metric. 1,200 documents created a representative distribution of entities that were comparable to the 5,568,714 training and test document set.

A team of 3 human annotators used a tool called Watson Knowledge Studio to annotate text as 1 of 13 entity types. The documents were pre-annotated from 10 dictionaries that searched for words and automatically created an initial annotation. The annotators corrected pre-annotations while adding others that were missed. Each day, the team met to discuss their kappa statistic or agreement score between each other over all entity types. Over a span of 3 weeks, the team produced a statistical entity model with a precision of 79%, recall of 73% and an F1 score of 76%. Even with a 14% entity word density over all documents, the overall annotator agreement score was at 70% with the majority of differences being the omission of a few words in a phrase.

4.3 Document2Vector

For each queried document, a summarization of keywords, concepts and entities from the document is unioned into feature vector s_i as shown in Equation 7. A doc2vec model was trained on 50 documents for each day of the top 300 players from each of the previous Fantasy Football seasons. In total, the training set included 94 GB of text. A second precise oriented doc2vec model was trained on 10 football dictionaries. In total, both models learned millions of vocabulary words during the training stage. The feature vector is input into both of the broad word embedding model and narrow encyclopedia model for a spatial word embedding summarization of the document. As shown in Equation 8, the full meaning around a player was the average spatial representation of all the documents for a particular player from the encyclopedia embedding, w_e, and the broad embedding, w_b.

$$\overline{s_i} = \overline{k_d} \cup \overline{c_d} \cup \overline{e_d} \quad (7)$$

$$\overline{a_p} = \frac{1}{N}\sum_{l=0}^{N} w_e\,(s_l) \cup w_b(s_l) \quad (8)$$

4.4 Deep Player Classification

Every player was probabilistically classified as a weekly boom, bust, play with hidden injury, or play meaningful touches. Each player state used an identical neural network topology but different weights. The topology includes 98 layers where 6 sequential paths are merged into a series of densely connected neurons. Every parallel sequence leveraged a dropout layer at the beginning to prevent overfitting the gradients. With the large number of layers, batch normalization was used in each parallel path to speed up training.

Through experimentation, the play with hidden injury and bust classifiers used the tanh activation function from Equation 9 while the play meaningful touches and breakout classifiers implemented the reLu activation function from Equation 10. The last layer uses a sigmoid activation function shown in Equation 11 to scale the output between 0 and 1. The networks used stochastic gradient decent to minimize the binary cross entropy. For each of the measurable player states such as boom and bust, a deep neural network has been designed and trained, dnn_{state}. Equation 12 shows the union of feature vectors average word embedding, $\overline{a_p}$, player bio vector, $\overline{b_p}$, and the social sentiment measures of a player, $\overline{s_p}$, discussed in section 3.6, are fed forward into each deep neural network for a probabilistic measurement.

$$a_{tanh}(z) = \frac{1-e^{-2(z)}}{1+e^{-2(z)}} \quad (9)$$

$$a_{relu}(z) = log(1+e^{x}) \quad (10)$$

$$a_{sigmoid}(z) = \frac{e^{z}}{e^{z}+1} \quad (11)$$

$$dnn_{state}(\overline{a_p} \cup \overline{b_p} \cup \overline{s_p}) = l_{96}\left(l_{95}\left(l_{\dots}\left(l_0(\overline{a_p} \cup \overline{b_p} \cup \overline{s_p})\right)\right)\right) \quad (12)$$

The inputs into the neural network include the player biographic, word 2 vector outputs, and social sentiment of each entity type in the document set. The output of each neural network learned the relationships between the input vectors.

4.5 Sentiment Analysis

Each of the pieces of player focused evidence within text form were labeled as positive, neutral or negative. However, labeling fantasy football text is challenging because:

- Sports articles typically talk about multiple players
- Player position impacts text meaning. For example, touchdown is good for a wide receiver but usually bad for a defense
- Agreement between human annotators is low in articles that are neutral.

Our system used a transformer model with a linear classification layer to classify the article sentiment. The sentiment model was trained using a pretrained transformer model based on OpenAI's Generative Pretrained Transformer Model-2 architecture (GPTM), which was trained on Wikitext-103 data [8]. We fine-tuned the model with our own labeled data by training the linear classifier within the GPTM model and altering hyperparameters.

Our training corpus included 900 sports articles that were labeled for players spanning different positions such as quarterback, kicker, tight end, running back, wide receiver, and defense. A corpus of domain specific words such as "pick-six" were added to the training data. During model training and apply, the system isolated the article text to the window of the player mentions. Articles where the player was only mentioned only once were excluded to increase the probability that the article was focused around a specific player. Individual players and specific teams can become associated with positive or negative sentiment in the text corpora. To negate this form of bias, we masked the player and team names during training.

After the sentiment classifier was trained, the Local Interpretable Model-Agnostic Explanations (LIME) framework was used to understand the model's predictions and behavior [21]. The LIME method generates a local linear approximation of the model and perturbs the model to identify features that influence the classification results the most. LIME was useful to understand the the relation between a model's word association and the predicted label. For example, "turnover" has to be interpreted as negative for

the offense and positive for the defense. LIME shows if "turnover" is weighted correctly within the context of offense and defense.

4.6 Player Score Probability Distribution

A multiple regression ensemble based on player position provided a point projection for each player. Equation 13 shows the general linear regression used for each position.

$$s(\bar{x}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_{|x|} x_{|x|} \quad (13)$$

$$pdf_j(h_o \ldots h_n \cup s_0 \ldots s_n) = min_{loss}(pdf_0, pdf_1, \ldots, pdf_{25}) \quad (14)$$

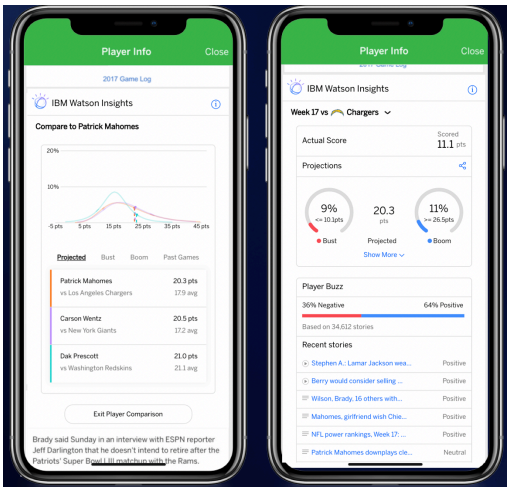


Figure 3: Mobile application player compare experience

Figure 3 shows a mobile user interface within the ESPN ecosystem that we created for Fantasy Football players. To produce the best PDF, the end of the machine learning pipeline fit 24 different PDFs to a player's historical and predicted performance as shown in Equation 14. Some example distributions include alpha, anglit, beta, bradford, chi, wald, vonmises, normal, and Rayleigh. If the player, such as a rookie, did not have enough historical data, similar player data was retrieved.

The distribution that fit the data the best was selected to run 1,000 simulations or random draws. The simulations produced more likely real-world curves for player performances. We highlighted the 15th and 85th percentile on the graph so that users could easily compare players.

5 Results

Overall, the system provided informative and accurate Fantasy Football insights from text, video, audio, and statistics. The system projected 88.2% of players to be within 10 points of their projection and 71% of player scores to be within 7 points of a projection. From a score distribution perspective, 83% of players are within the high score range while 71% of players are within the low score range. Impressively, 90% of players either boomed or were close to boom when predicted to boom. On the other end, 78% of players either busted or were close to a bust when predicted to bust.

5.1 Document2Vector Results

The model was tested with two different types of semantic meaning evaluations. First, an analogy test was provided to the model. If the relation Travis Kelce is to the Chiefs as Todd Gurley is to the X is presented to the model, the correct answer for X should be the Rams. In the player to team analogy testing, the correct answer was in the top 1% of the data 100% of the time. The team to location analogy was slightly lower, with a 93.48% accuracy because the natural queries were not focused around teams. The second test provided a set of keywords to the model and expected a related word. For example, if Tom Brady input into the model, we would expect to see the Patriots as output.

Test	Subject	Criteria	Accuracy
Analogy	Players:Team	Top 500 (<1% of the data)	100%
Analogy	Team:Location	Top 500 (<1% of data)	93.48%
Keyword	Players	Top 70	80%
Keyword	Team & Location	Top 500	74%

Table 1: Document2Vector Tests and Results

5.2 Deep Learning Results

The bust game classifier had an accuracy of 55% with a modest class separation, while the boom classifier had an accuracy of 67%. The bust classifier was optimized on real world player bust distribution and accuracy because players with high bust probabilities significantly over scored their projections on average. The bust players that were missed and marked incorrect were very close to the binary threshold of 0.5. Further, the negative predictive value of the bust model is 85.5% accurate and it produces a real-world percentage of bust players at 12%. As a tradeoff, the over predicting of busts would be worse than a high accuracy. The accuracy number is not as meaningful an evaluation metric as the negative predictive value and percentage of players predicted to be a bust.

The play with injury classifier had an accuracy of 77% with a positive predictive value of 68.1%. The positive predictive value is very important for this classifier so that we know if a player is going to play with a hidden injury. The play meaningful minutes model produced an accuracy of 91.4%. The output of the class and probability provide valuable predictors for the score projections as well as insights about each football player.

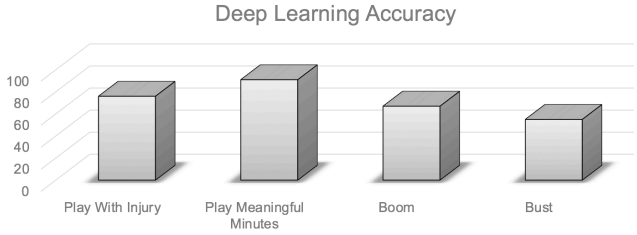
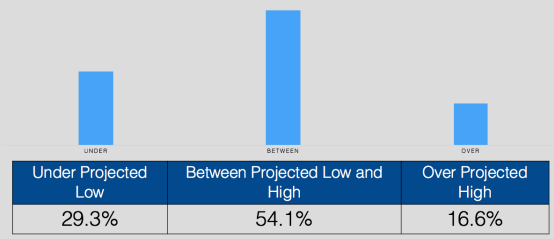


Figure 4: Deep Learning Results

From a real-world distribution of players that boom or bust, we were close to our objectives. Between 12-16% of players generally boom while 30% can bust week over week. Figure 4 shows our

results. Fantasy football users would quickly lose confidence in our system if we over predicted boom or bust.



Under Projected Low	Between Projected Low and High	Over Projected High
29.3%	54.1%	16.6%

Figure 5: Boom and Bust Player Distribution

In addition, we ensured that players on more popular teams were not biased with our deep learning algorithms. We examined the Generalized False Positive Rate (GFPR) and the Generalized False Negative Rate (GFNR) for both boom and bust between privileged and unprivileged values. After post processing, the bias within boom and bust models were close to zero for GFPR and GFNR metrics while not impacting model accuracy.

5.3 Sentiment Results

The sentiment classifier had an accuracy of 81%. The classifier did well on labelling positive and negative articles with accuracies of 87% and 83%. The neutral sentiment had an accuracy of 61%, which was harder to predict because of inter-annotator disagreement. The sentiment labels next to each piece of evidence provided users with overall estimation of the crowd's opinion of each player.

5.4 Score Projection Results

A linear combination of deep learning player states and ESPN statistical data produced the best RMSE score of 6.78. On average, each player projected to score significant points over all positions will have a projection score that is off by 6.78 points when ESPN and our system is combined.

Model	RMSE (Point Error)
ESPN Projection	6.81
Watson Adjusted Projection	6.92
Combined Projection	6.78

Table 2: Point Projection Performance

The accumulation of the score projections over the duration of the football season provide data points for curve fitting as discussed in section 3.5. The low RMSE validates the probability density curve fits so that users can compare player shapes to each other as shown in Figure 3.

6 Deployment on Cloud

After training and evaluating all of the models and mathematical techniques illustrated in Section 5, we uploaded them to COS. The binary files were placed into buckets so that the deployed applications could pull down the models to local disk. The models were only pulled before each 300-player batch job. In addition, we could force each phase of the machine learning pipeline to pull down a new model if it was updated. This approach gave the team the flexibility to test new model architectures and hyperparameters throughout the long fantasy football season.

The use of the double CDN with the JSON content generation pattern discussed in Section 3, shielded the project from any backend processing inaccuracies. The diversity of data at different volumes can cause machine learning resource contention or unanticipated data states. For example, a few of the probability density function curve fittings produced asymptotes that confused users. As a result, we were able to remove a type of curve fit and rerun a specific player without any down time.

The flexibility of Cloud Foundry and Docker containers helped to support the movement of cloud workloads vertically and horizontally. The team could rapidly deploy changes, hot fixes, or new features in an agile environment without having to wait for long build times.

7 Application Impact

Each week throughout 2017, 2018 and 2019, fantasy football team owners had the option of using our system to set team lineups. This was presented to fans in a few different forums such as player screens in the ESPN Fantasy App and segments aired on TV or broadcasted on national radio. We found that empirical based decisions supported by the system help to minimize the temptation to make biased sit and start decisions concerning favored and unfavored players . In the first month of 2019 alone, over 5.5 billion insights were produced for the 9.8 million users that accessed the ESPN Fantasy App for 2.4 billion minutes that month. This unprecedented level of depth and insight from unstructured data complimented by ESPN's traditional player statistics and analysis provided a comprehensive and detailed story about each player.

Towards the end of the 2018 ESPN Fantasy Football season, over a thousand players participated in a survey to measure the impact of our system. From the active survey respondents, over 80% of the users who utilized the feature said the Watson AI insights generated from our system helped them to enjoy fantasy football better. The more a fan followed the NFL, the more likely they used our system.

From a marketing perspective, we had 2 celebrity-focused fantasy football leagues that promoted our system. Former NFL players, NBA players, ESPN on-air talent, IBM data scientists, and a movie star were among the competitors in a public ESPN influencer league. The influencer league and interest around our system generated 35 million impressions, 10.6 million video views, and 744 thousand total engagements from over 600 media pieces. The conversation on social media about ESPN Fantasy Football grew 24% in positivity from 2017.

Figure 6: 2018 and 2019 Fantasy Football Social Influencer League

Source	Link
Explainer Video	https://youtu.be/xCszkWFAxmA
IBM Homepage	https://www.ibm.com/sports/fantasy
ESPN Fantasy Show	https://youtu.be/4BsDzKvBb3E
Blog Series	https://developer.ibm.com/series/watson-behind-the-code-fantasy-football-2018/
Front of Code Commercial	https://www.youtube.com/watch?v=1cYrk67I00E
2019 Podcast	https://ibm.co/2Sv5Uud
2018 Podcast	https://bit.ly/2BQ3PS6

Table 3: Additional Impact Information

8 Future Work

To further our research and user experience using multimedia data throughout fantasy football, we are developing a player trade discoverer. The trade discoverer will find possible trading partners within a league and suggest a trade. We would like to examine the tradeoff between the likelihood of a trade being accepted with the utility gained by the initiating team. Our goals are to increase successful trades throughout fantasy football that help all teams involved within a transaction to increase their team's score ceiling. The experience will be engaging and insightful across mobile and desktop applications.

From a deployment perspective, the fantasy football system will be deployed on OpenShift for the 2020 season. OpenShift is a Docker container management system that runs on top of Kubernetes. With a total of 16 projects, 7 will be wrapped into Docker images for deployment on an OpenShift cluster. The build pipeline will automatically handle code changes in GitLab, create new Docker images, and deploy the applications across Kubernetes pods in a canary strategy.

ACKNOWLEDGMENTS

We would like to thank ESPN for their support by opening their television studios to our project. In addition, IBM Marketing, including Noah Syken, John Kent, Elizabeth O'Brien, and Kristi Kolski provided feedback and encouragement. We thank Jeffery Gottwald for his designs. We thank Stephania Bell, Field Yates, Daniel Dopp, Matthew Berry, Charles Woodson, Bonnie Bernstein, John Urschel, Baron Davis, Justin Tuck, Mike Greenberg, and Jerry Ferarra for using our platform within celebrity leagues.

REFERENCES

[1] A. Narayanan, "Translation tutorial: 21 fairness definitions and their politics," Conference on Fairness, Accountability, and Transparency, February 2018.

[2] Akbik, A., Blythe, D., & Vollgraf, R. (2018). Contextual String Embeddings for Sequence Labeling. COLING.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[4] Baevski, Alexei & Edunov, Sergey & Liu, Yinhan & Zettlemoyer, Luke & Auli, Michael. (2019). Cloze-driven Pretraining of Self-attention Networks.

[5] D. Yao, J. Bi, J. Huang, and J. Zhu, "A word distributed representation based framework for large-scale short text classification," in *IEEE IJCNN*, 2015, pp. 1-7.

[6] G. Dzodom and F. Shipman, "Data-Driven Web Entertainment: The Data Collection and Analysis Practices of Fantasy Sports Players," in *Proc. WebSci*, Bloomington, IN, June 23-26, 2014, ACM 978-1-4503-2622-3/14/06.

[7] Hermann, E and N. Adebia, "Machine Learning Applications in Fantasy Basketball.", semantic scholar, 2015.

[8] HuggingFace. 2019. NAACL Transfer learning tutorial. (Jun 2019). Retrieved Dec 21, 2019 from https://github.com/huggingface/naacl_transfer_learning_tutorial

[9] L. Kaiser, A. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, "One Model To Learn Them All," arXiv:1706.05137v1 June 16, 2017.

[10] Landers, J. and B. Duperrouzel, "Machine Learning Approaches to Competing in Fantasy Leagues for the NFL." IEEE Transactions on Games, vol. 11, issue 2, 2019.

[11] M. Du, F. Yang, N. Zou and X. Hu, "Fairness in Deep Learning: A Computational Perspective," https://arxiv.org/abs/1908.08843, Aug. 2019.

[12] N. Dunnington, "Fantasy football projection analysis," Ph.D. dissertation, Depart. Econ., Univ. Oregon, Eugene, OR, USA, 2015.

[13] P. Wang, B. Xu, J. Xu, G. Tian, C. Liu, H. Hao, "Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification," Neurocomputing, Vol. 174, Part B, 01/22/2016, pp. 806-814.

[14] Peters, Matthew & Neumann, Mark & Iyyer, Mohit & Gardner, Matt & Clark, Christopher & Lee, Kenton & Zettlemoyer, Luke. (2018). Deep contextualized word representations.

[15] R. Bellamy, K. Dey, M. Hind, S. Hoffman, S. Houde, K. Kennan, P. Lohia, J. Martino, S. Mehta, A. Mojsilvoic, S. Nagar, K. Ramamurthy, J. Richards, D. Saha, P. Sattigeri, M. Singh, K. Varshney and Y. Zhang, "AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias," https://arxiv.org/abs/1810.01943

[16] R. Binns, "Fairness in Machine Learning: Lessons from Political Philosophy," Conference on Fairness, Accountability, and Transparency, February 2018.

[17] R. Collobert and J. Weston, "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning," in Proc 25th International Conference on Machine Learning, Helsinki, Finland, 2008.

[18] R. Lutz, "Fantasy Football prediction," arXiv:1505.06918, 2015.

[19] Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D. & Sutskever, I. (2018), 'Language Models are Unsupervised Multitask Learners', .

[20] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparsetransformers.arXiv preprint arXiv:1904.10509, 2019.

[21] Ribeiro Marco Tulio Correia, LIME, (2016). GitHub repository, Retrieved Dec 21, 2019 https://github.com/marcotcr/lime

[22] S. Hirsh, C. Anderson, and M. Caselli, "The Reality of Fantasy: Uncovering Information-Seeking Behaviors and Needs in online Fantasy Sports," in *Proc. CHI*, Austin Texas, May 5-10, 2012, ACM 978-1-4503-1016-1.

[23] S. Jansen, "Word and Phrase Translation with word2vec," arXiv:1705.03127, 2017.

[24] Seal, C., "Can machine learning help improve your fantasy football draft?", https://medium.com/fantasy-outliers/can-machine-learning-can-help-improve-your-fantasy-football-draft-4ceea1f1b2bd, 07/07/2019.

[25] T. Matthews, S. D. Ramchurn, and G. Chalkiadakis, "Competing with humans at fantasy football: Team formation in large partially-observable domains," in *Proc. AAAI*, 2012, pp. 1394–1400.

[26] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," Advances in neural information processing systems, pp. 3111-3119, 2013.

[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv preprint arXiv:1302.3781, 01/16/2013.

[28] X. Glorot, A. Bordes, Y. Bengio, "Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach," in Proc. *ACM ICML*, 2011, pp. 513-520.