Predicting your hero's next move

An analysis of League of Legends data

Michele Zanitti

Main objective:

Can we accurately predict where players move in League of Legends?

The journey

- Understand the data
- descriptive statistics and tests
- Prediction of win/lose
- Prediction of the next movement

Exploring the data

Two families of datasets (JSON):

- MongoDB sequences -> for Keras and statistics
- API Riot -> data enrichment

Exploring the data - Format

Keras sequences: array in pickle format

Enrichment: JSON in pickle format to pandas dataframe

Exploratory Descriptive Analysis

Analysis run on two fronts:

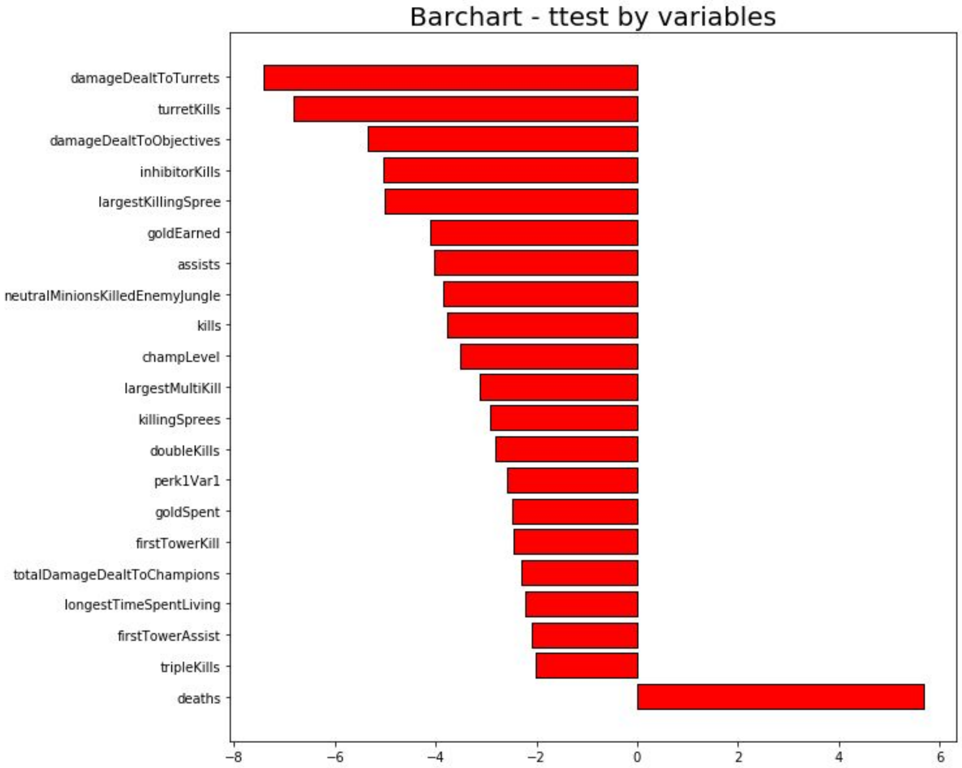
- Global
- Single match

Statistical approaches:

- Difference in means
- Heatmaps

Dataframe used: Game and Timeline

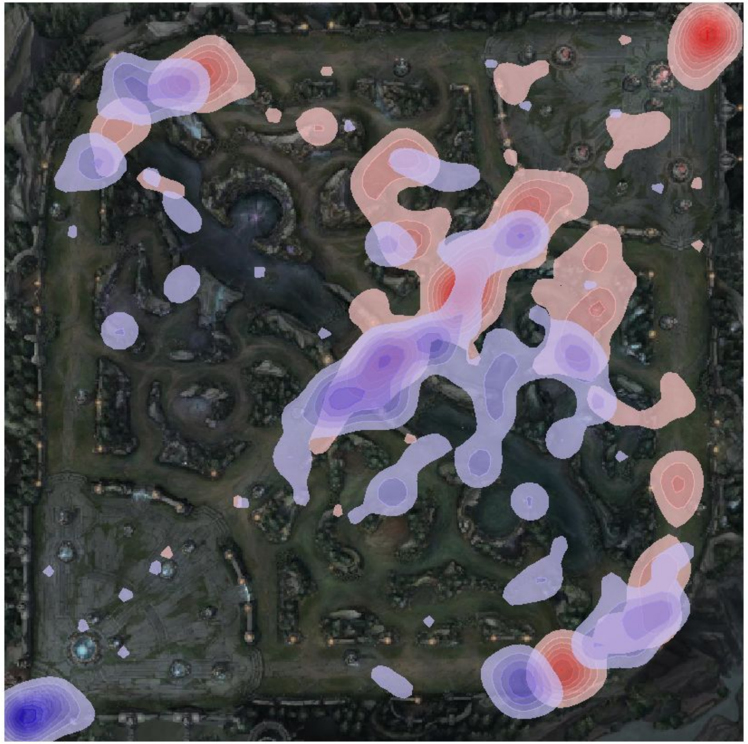
Global statistics



On 100 variables, grouped by team (winner or loser), only 18 proved to be significatively different in means

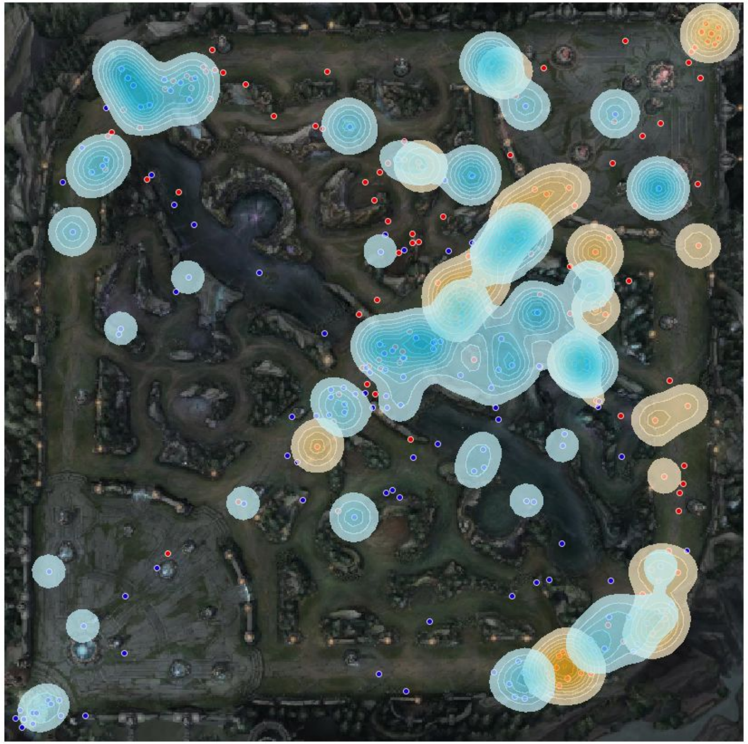
(confidence level = 0.05)

Single match statistics



Heatmap: difference in density in spatial distribution computed with kernel methods

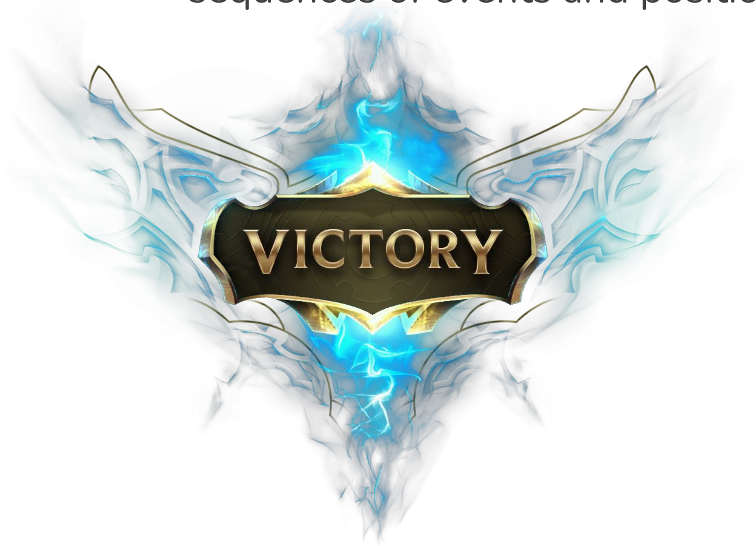
Single match statistics



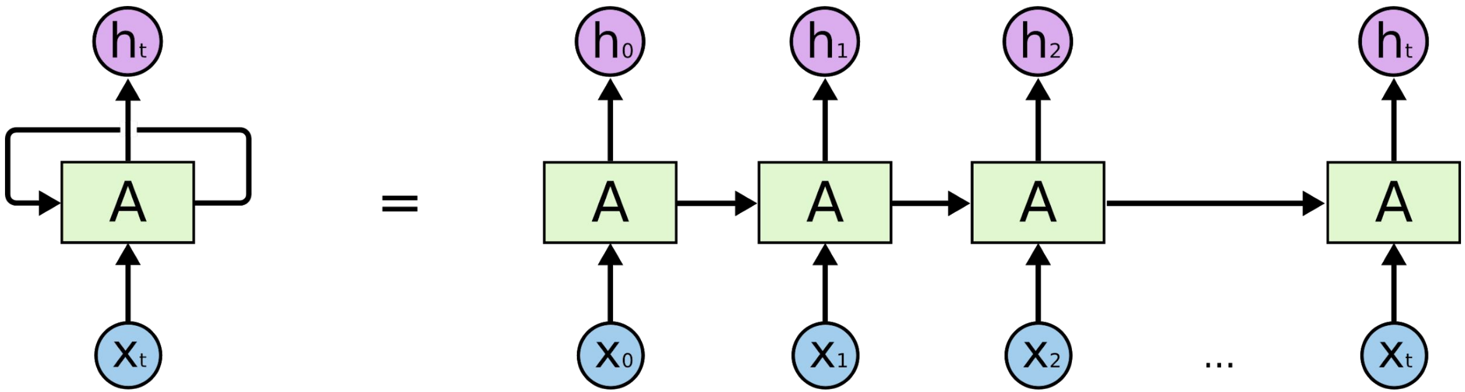
Example of a density distribution linked to the event “ward placed”

Match outcome prediction

Two models (RNN and LSTM) tested to predict victory or defeat, given the sequences of events and positions of each player.

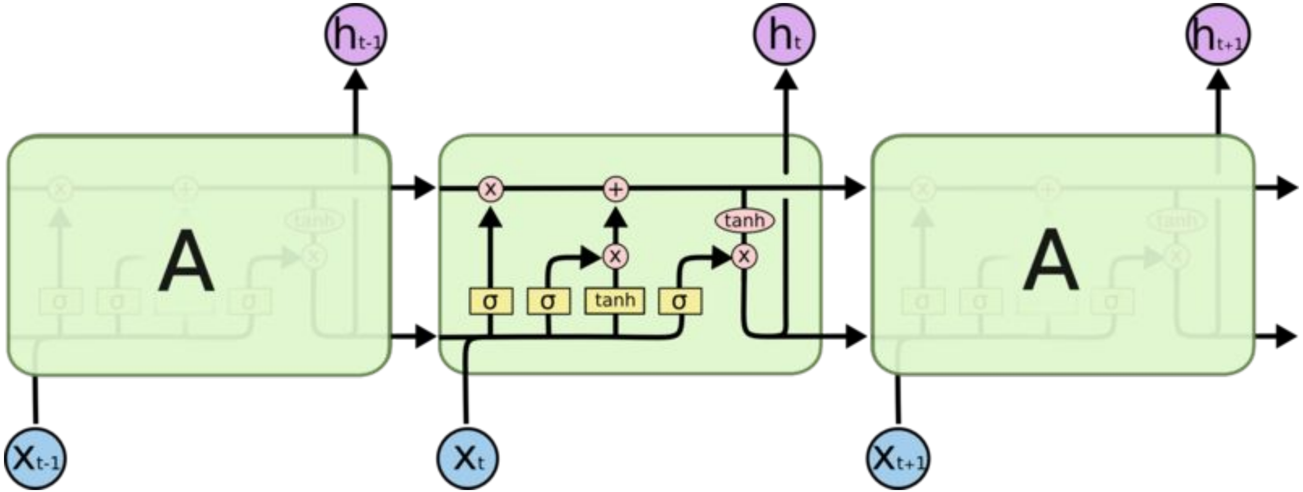


The model: RNN



Criticality: vanishing gradient for long sequences

The model: LSTM



Forget, remember, output

Sequence creation

Events and position are encoded in order to be processed as a one-dimensional sequence:

- Events transformed into numbers (es. 10001=building kill)
- Sets of position coordinates (x,y) transformed from a 15000x15000 to 100x100 matrix
- Multidimensional to one-dimensional

Model and Parameters

Which model has better accuracy?

Validation parameters:

```
Epochs = 8
batch size = 15
Validation split = 20%
```

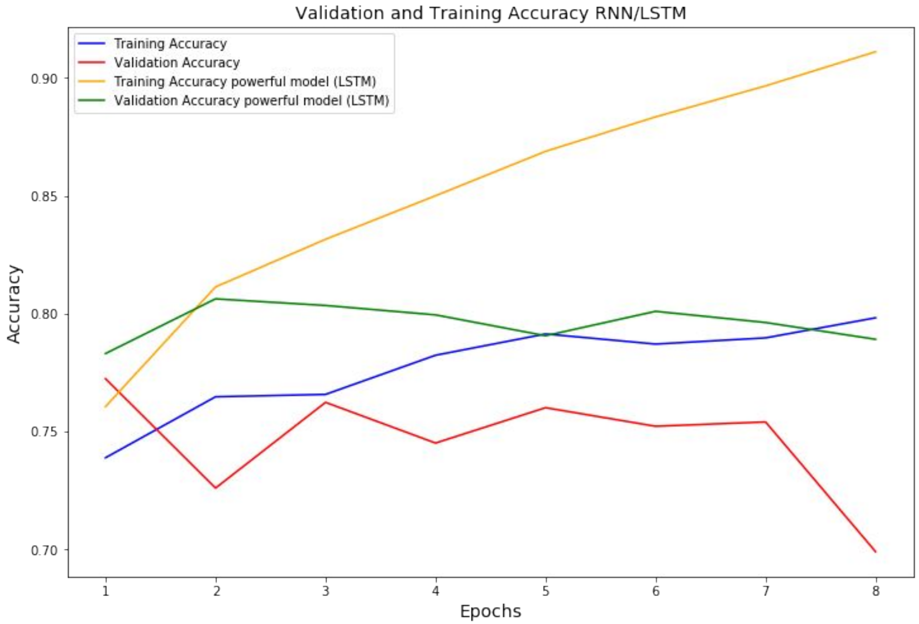
RNN parameters:

```
optimizer='rmsprop',
loss='binary_crossentropy'
```

LSTM parameters:

```
neurons = 50
neuronsHL = 70
dropOut_rate = 0.4
```

Outcome prediction accuracy



LSTM wins!

Improvement of ~10% over RNN

Still huge gap between training and validation accuracy scores.

Predict the next move

Predict the next move given the three precedent sets of positions

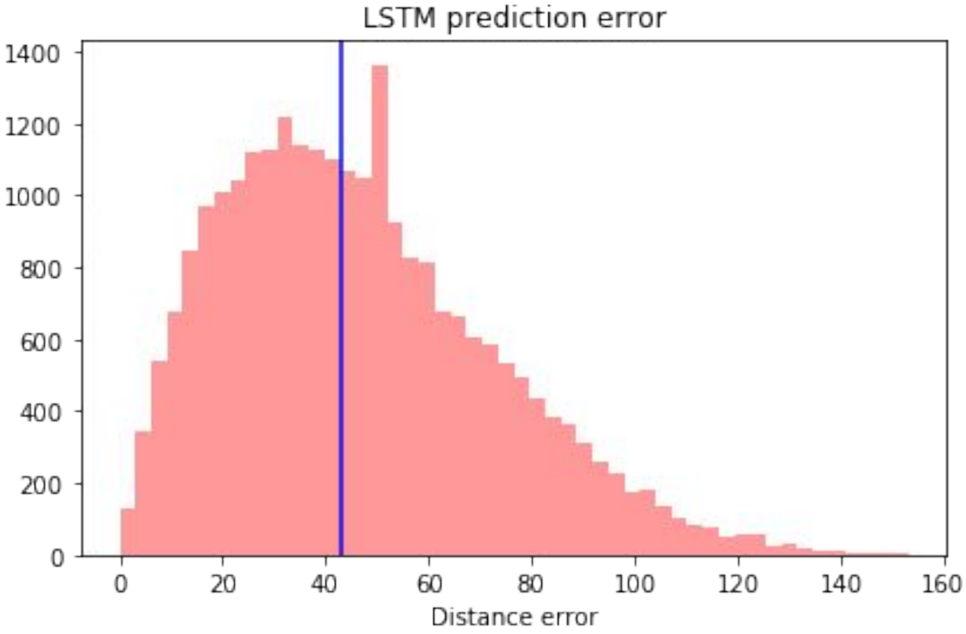
Model

- LSTM model
- Chebyshev distance

```
model = Sequential()
model.add(LSTM(200,
               activation='relu', # linear activation
               input_shape=(n_input, n_features),  # 3 -> 1
               recurrent_dropout=0.2, # rnn regularization to avoid
overfit
               kernel_initializer='normal')) # initialization

model.add(Dense(1))
model.compile(optimizer='adamax', # very precise optimizer
              loss='mse') # more weight to larger differences
```

Results



On 250.000 position data analyzed

average distance error is 28%

median = 42

std deviation = 26.3

Criticalities and comments

- Time between frames is too wide (1 minute)

- Poor optimization of Chebyshev distance implies long computing time (low percentage of data analyzed)

- Low accuracy in predicting special actions (deaths, teleports and backs)

- Potential improvement with a change of kind of neural network and more computational power

Conclusion

- Connection between games, users and avatars
- Understanding winning behaviors, movements and events
- Predict win or lose with confidence
- Predict player’s next movement

Thank you!