In [1]:
```
import pandas as pd

import re
from emoji import UNICODE_EMOJI
from textblob import TextBlob
import altair as alt
import numpy as np
from collections import Counter
import string

import nltk
nltk.download('vader_lexicon')
nltk.download('brown')
nltk.download('punkt')
nltk.download('stopwords')

from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/jovyan/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package brown to /home/jovyan/nltk_data...
[nltk_data]   Package brown is already up-to-date!
[nltk_data] Downloading package punkt to /home/jovyan/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/jovyan/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

The data cleaning/manipulation functions

In [2]:

```
def extract_tags(text):
    return re.findall("#([a-zA-Z0-9_]{1,50})", text)

def extract_emoji(text):
    return [ch for ch in text if ch in UNICODE_EMOJI['en']]


def clean_tweet(txt):
    temp = re.sub("@[A-Za-z0-9_]+","", txt)
    temp1 = re.sub("#[A-Za-z0-9_]+","", temp)
    temp2 = re.sub(r"http\S+", "", temp1)

    result=''.join(i for i in temp2.lower() if (i.isalpha() or i==' '))
    return result

def word_list(tweet):

    lst = word_tokenize(tweet)
    lst1 = []
    stops = list(stopwords.words('english'))
    for w in lst:
        if w not in stops:
            lst1.append(w)

    return lst1

def sentiment(tweet):
    blob = TextBlob(tweet)

    return blob.sentiment.polarity


def get_date(date):

    return date[:10]

def get_hour(date):

    return date[11:13]
def get_10min(date):

    return date[14]+'0'

def get_min(date):
```

```
    return date[14:16]

def firm_pos(score):
    if score >= 0.7:
        return 1
    else: return 0

def pos(score):
    if (score >= 0.25) & (score < 0.7):
        return 1
    else: return 0

def neutral(score):
    if (score >= -0.25) & (score < 0.25):
        return 1
    else: return 0

def neg(score):
    if (score > -0.7) & (score < -0.25):
        return 1
    else: return 0

def firm_neg(score):
    if score <= -0.7:
        return 1
    else: return 0
```

Import data, check duplicate or missing value, remove rows if exists

In [3]:
```
df= pd.read_csv('Project Data/2020 world series.csv')
df['id'].duplicated(keep='last').sum()
```

Out[3]:
```
0
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
id      0
date    1
text    1
dtype: int64
```

In [5]:
```
df = df.dropna().reset_index()
df.drop(columns=['index'],inplace=True)
```

Apply data cleaning/manipulation techniques on the data, we now have the used words, tags, emojis, sentiment score, and specific date/hour/min data.

In [6]:
```
df['tags']= df.apply(lambda row: extract_tags(row['text']), axis=1)
df['emojis']= df.apply(lambda row: extract_emoji(row['text']), axis=1)
df['clean_text']= df.apply(lambda row: clean_tweet(row['text']), axis=1)
df['words']= df.apply(lambda row: word_list(row['clean_text']), axis=1)
df['sentiment_score']= df.apply(lambda row: sentiment(row['clean_text']), axis=1)
df['day']= df.apply(lambda row: get_date(row['date']), axis=1)
df['hour']= df.apply(lambda row: get_hour(row['date']), axis=1)
df['10min']= df.apply(lambda row: get_10min(row['date']), axis=1)
df['min']= df.apply(lambda row: get_min(row['date']), axis=1)
df['POS']= df.apply(lambda row: firm_pos(row['sentiment_score']), axis=1)
df['pos']= df.apply(lambda row: pos(row['sentiment_score']), axis=1)
df['neu']= df.apply(lambda row: neutral(row['sentiment_score']), axis=1)
df['neg']= df.apply(lambda row: neg(row['sentiment_score']), axis=1)
df['NEG']= df.apply(lambda row: firm_neg(row['sentiment_score']), axis=1)

df.head()
```

Out[6]:

	id	date	text	tags	emojis	clean_text	words	sentiment_score	day	hour	10min	min	POS	pos	n
0	1317978642094305282	2020-10-18 23:59:29+00:00	My ideal rotation for today:\n\nMay (3 innings...	[WorldSeries, GoDodgers]	[]	my ideal rotation for todaymay innings gonsol...	[ideal, rotation, todaymay, innings, gonsolin,...	0.9	2020-10-18	23	50	59	1	0	
1	1317978309360320514	2020-10-18 23:58:10+00:00	Rays fans, who do y'all want the #Rays to face...	[Rays, WorldSeries, Braves, Dodgers]	[]	rays fans who do yall want the to face in thi...	[rays, fans, yall, want, face]	0.0	2020-10-18	23	50	58	0	0	
2	1317978244277297152	2020-10-18 23:57:54+00:00	Respect for the Dodger blue. The Houston Astro...	[WorldSeries, BoysOfSUMMER, Brooklyn, goodreads]	[]	respect for the dodger blue the houston astros...	[respect, dodger, blue, houston, astros, cheat...	0.5	2020-10-18	23	50	57	0	1	
3	1317977949878931457	2020-10-18 23:56:44+00:00	#LATogether go to the #WorldSeries @Dodgers	[LATogether, WorldSeries]	[]	go to the	[go]	0.0	2020-10-18	23	50	56	0	0	
4	1317977838021115904	2020-10-18 23:56:18+00:00	Let's #MixItUp to the #WorldSeries tonight @Br...	[MixItUp, WorldSeries]	[]	lets to the tonight	[lets, tonight]	0.0	2020-10-18	23	50	56	0	0	

See the overall flow of tweet & sentiment

In [7]:
```
tweet_count = df.groupby(['day','hour']).size().reset_index()
tweet_count['date'] = tweet_count['day'] + ' ' + tweet_count['hour'] + ':00'
tweet_count.columns = ['day', 'hour', 'count', 'date']

tweet_count.head()
```

Out[7]:

	day	hour	count	date
0	2020-10-18	00	168	2020-10-18 00:00
1	2020-10-18	01	39	2020-10-18 01:00
2	2020-10-18	02	33	2020-10-18 02:00
3	2020-10-18	03	1011	2020-10-18 03:00
4	2020-10-18	04	694	2020-10-18 04:00

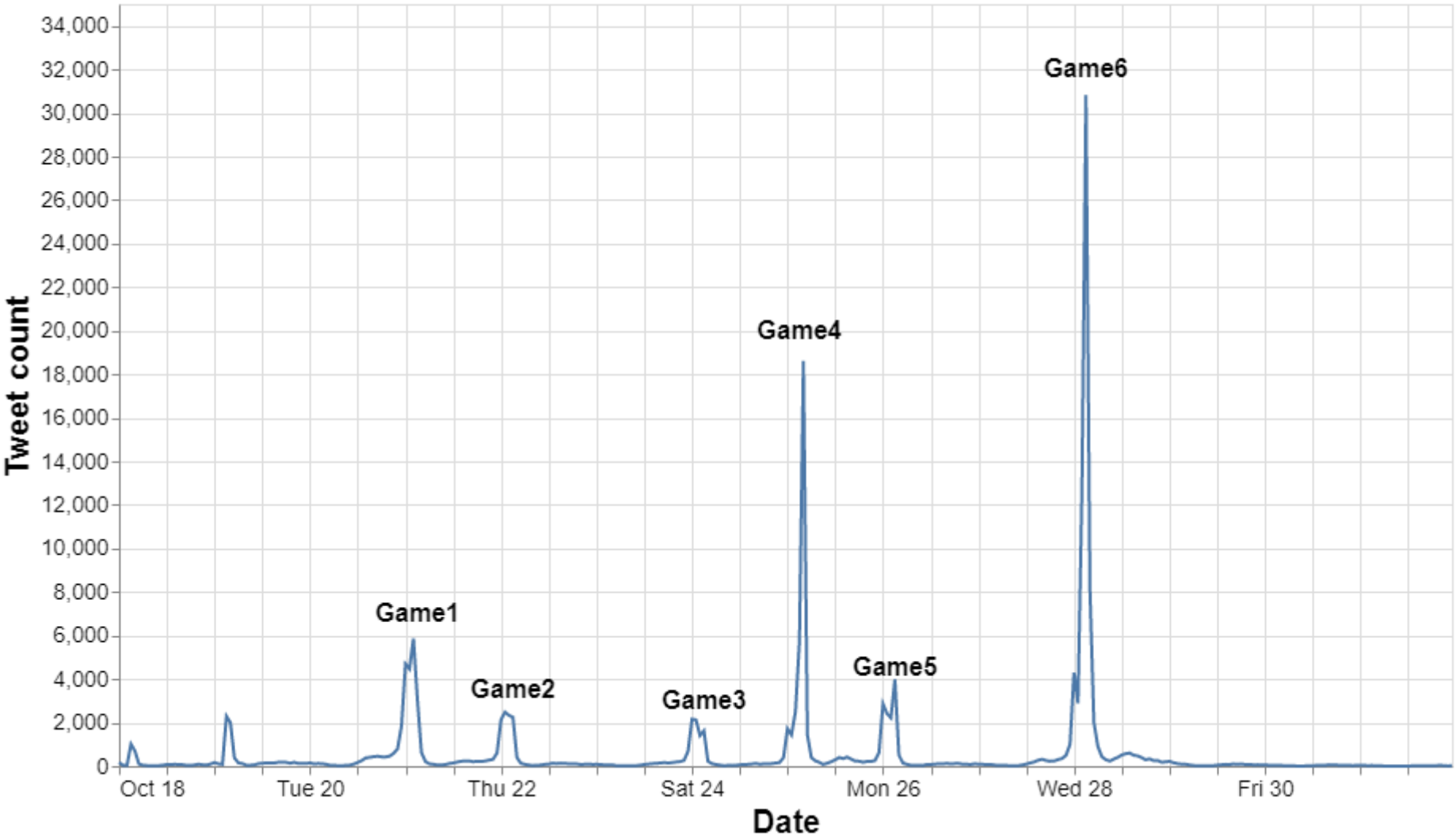
In [8]:

```
annotations = [['2020-10-21 03:00',7000, 'Game1'],
               ['2020-10-22 03::00',3500, 'Game2'],
              ['2020-10-24 03::00',3000, 'Game3'],
               ['2020-10-25 03::00',20000, 'Game4'],
              ['2020-10-26 03::00',4500, 'Game5'],
              ['2020-10-28 03::00',32000, 'Game6']]

a_df = pd.DataFrame(annotations, columns=['date','values','note'])

line = alt.Chart(tweet_count).mark_line().encode(
    x=alt.X('date:T',title='Date'),
    y=alt.Y('count:Q',title='Tweet count')
)

text=alt.Chart(a_df).encode(
    x=alt.X('date:T'),
    y=alt.Y('values:Q'),
    text='note').mark_text(size=16,fontWeight='bold')

(line + text).properties(width=840,height=480,title={
      "text": ["Tweets Count Flow - 2020 World Series"],
      "subtitle": ["Tweet count about the World Series on Twitter by hour"]
    }).configure_axis(
    labelFontSize=14,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 25,
    subtitleFontSize = 15
)
```

Out[8]:

Tweets Count Flow - 2020 World Series

Tweet count about the World Series on Twitter by hour



In [9]:
```
# compute the 10-hour rolling average percentage of each sentiment degree label

senti = df.groupby(['day','hour']).sum()[['POS', 'pos', 'neu','neg', 'NEG']]
senti = senti.reset_index()
senti['date'] = senti['day'] + ' ' + senti['hour'] + ':00'
senti['size'] = pd.Series(df.groupby(['day','hour']).size().values)
senti[['POS', 'pos', 'neu', 'neg', 'NEG','10hr_count']] = senti.rolling(window=10,min_periods=1).sum()[['POS', 'pos', 'neu', 'neg'
senti['POSITIVE'] = senti['POS'] / senti['10hr_count']
senti['positive'] = senti['pos'] / senti['10hr_count']
senti['neutral'] = senti['neu'] / senti['10hr_count']
senti['negative'] = senti['neg'] / senti['10hr_count']
senti['NEGATIVE'] = senti['NEG'] / senti['10hr_count']

senti.head()
```

Out[9]:

	day	hour	POS	pos	neu	neg	NEG	date	size	10hr_count	POSITIVE	positive	neutral	negative	NEGATIVE
0	2020-10-18	00	9.0	27.0	102.0	28.0	2.0	2020-10-18 00:00	168	168.0	0.053571	0.160714	0.607143	0.166667	0.011905
1	2020-10-18	01	9.0	36.0	129.0	31.0	2.0	2020-10-18 01:00	39	207.0	0.043478	0.173913	0.623188	0.149758	0.009662
2	2020-10-18	02	10.0	41.0	149.0	37.0	3.0	2020-10-18 02:00	33	240.0	0.041667	0.170833	0.620833	0.154167	0.012500
3	2020-10-18	03	112.0	238.0	817.0	78.0	6.0	2020-10-18 03:00	1011	1251.0	0.089528	0.190248	0.653078	0.062350	0.004796
4	2020-10-18	04	182.0	413.0	1215.0	125.0	10.0	2020-10-18 04:00	694	1945.0	0.093573	0.212339	0.624679	0.064267	0.005141

In [10]:
```
# create a dataframe for visualization

date = []
value = []
label = []

senti_flow = pd.DataFrame()

for i in ['POSITIVE', 'positive', 'neutral', 'negative', 'NEGATIVE']:
    lst=[]
    lst1=[]
    lst2=list(senti.date.values)

    for j in range(len(senti)):
        lst.append(i)
        lst1.append(senti[i][j])

    date += lst2
    value += lst1
    label += lst

senti_flow['date'] = pd.Series(date)
senti_flow['Sentiment_label'] = pd.Series(label)
senti_flow['perct'] = pd.Series(value)
senti_flow.head()
```

Out[10]:

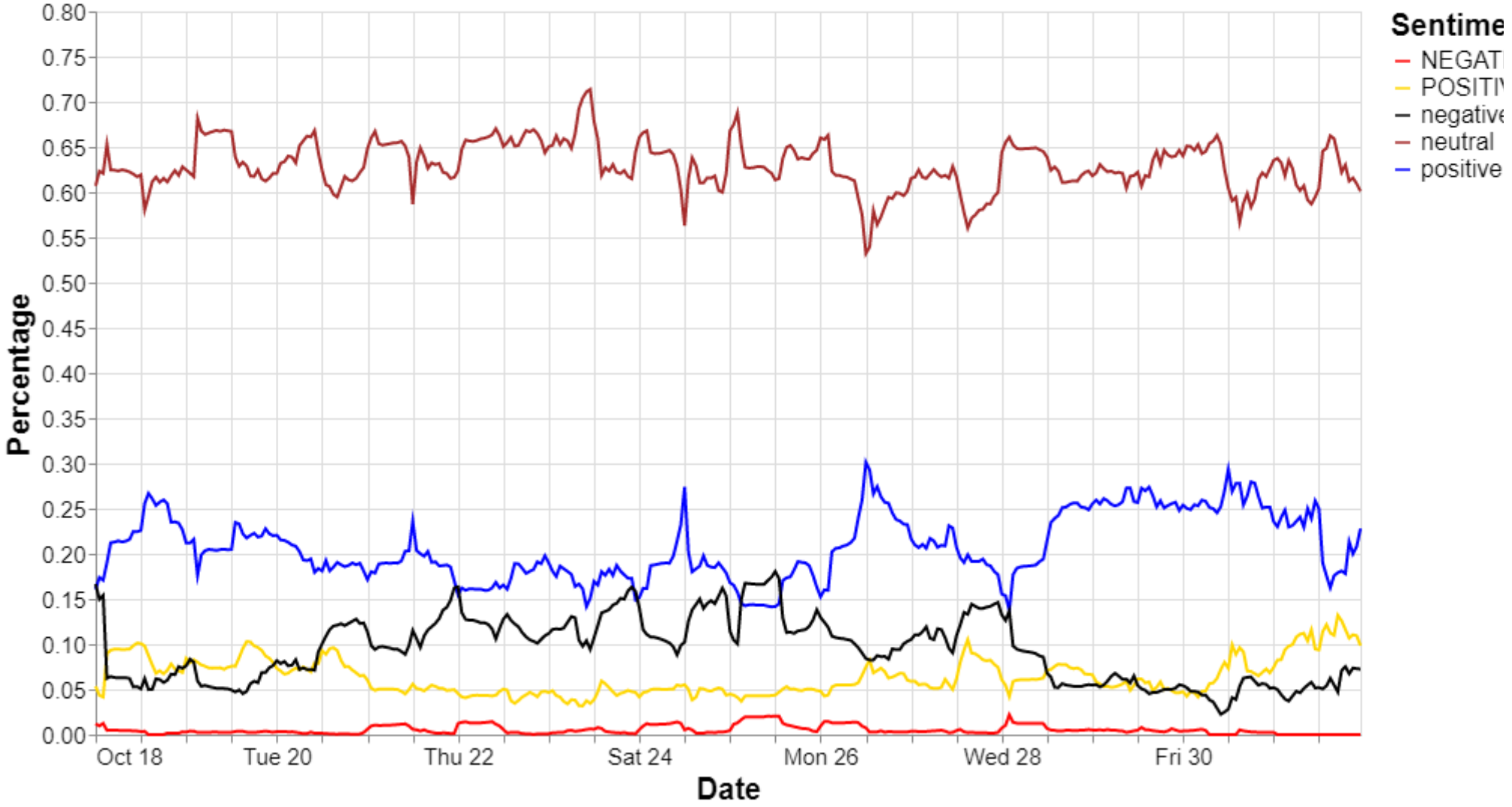
	date	Sentiment_label	perct
0	2020-10-18 00:00	POSITIVE	0.053571
1	2020-10-18 01:00	POSITIVE	0.043478
2	2020-10-18 02:00	POSITIVE	0.041667
3	2020-10-18 03:00	POSITIVE	0.089528
4	2020-10-18 04:00	POSITIVE	0.093573

In [11]:
```
alt.Chart(senti_flow).mark_line().encode(
    x=alt.X('date:T',title='Date'),
    y=alt.Y('perct:Q',title='Percentage'),
    color=alt.Color('Sentiment_label',
                    scale=alt.Scale(
            range=['red', 'gold','black','brown','blue']))
).properties(width=840,height=480,title={
      "text": ["Sentiment Flow - 2020 World Series"],
      "subtitle": ["The percentage flow of each degree of sentiment on Twitter, values are computed by the 10-hour rolling average
    }).configure_axis(
    labelFontSize=15,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 28,
    subtitleFontSize = 16
).configure_legend(
    titleFontSize=20,
    labelFontSize=16
)
```

Out[11]:

Sentiment Flow - 2020 World Series

The percentage flow of each degree of sentiment on Twitter, values are computed by the 10-hour rolling average



Emoji/tag/word

In [12]:
```
# this return the top 50 most common items in the columns (emoji/tag/word)

def top_item(data,label):

    lst = []
    for i in data[label]:
        lst += i

    C = Counter(lst)
    top50 = C.most_common(50)
    count_df = pd.DataFrame(top50,columns = [label,'count'])

    return count_df
```

In [13]:
```
c= top_item(df,'tags')
c

# the top 50 most frequently used tags within the 2020 world series
```

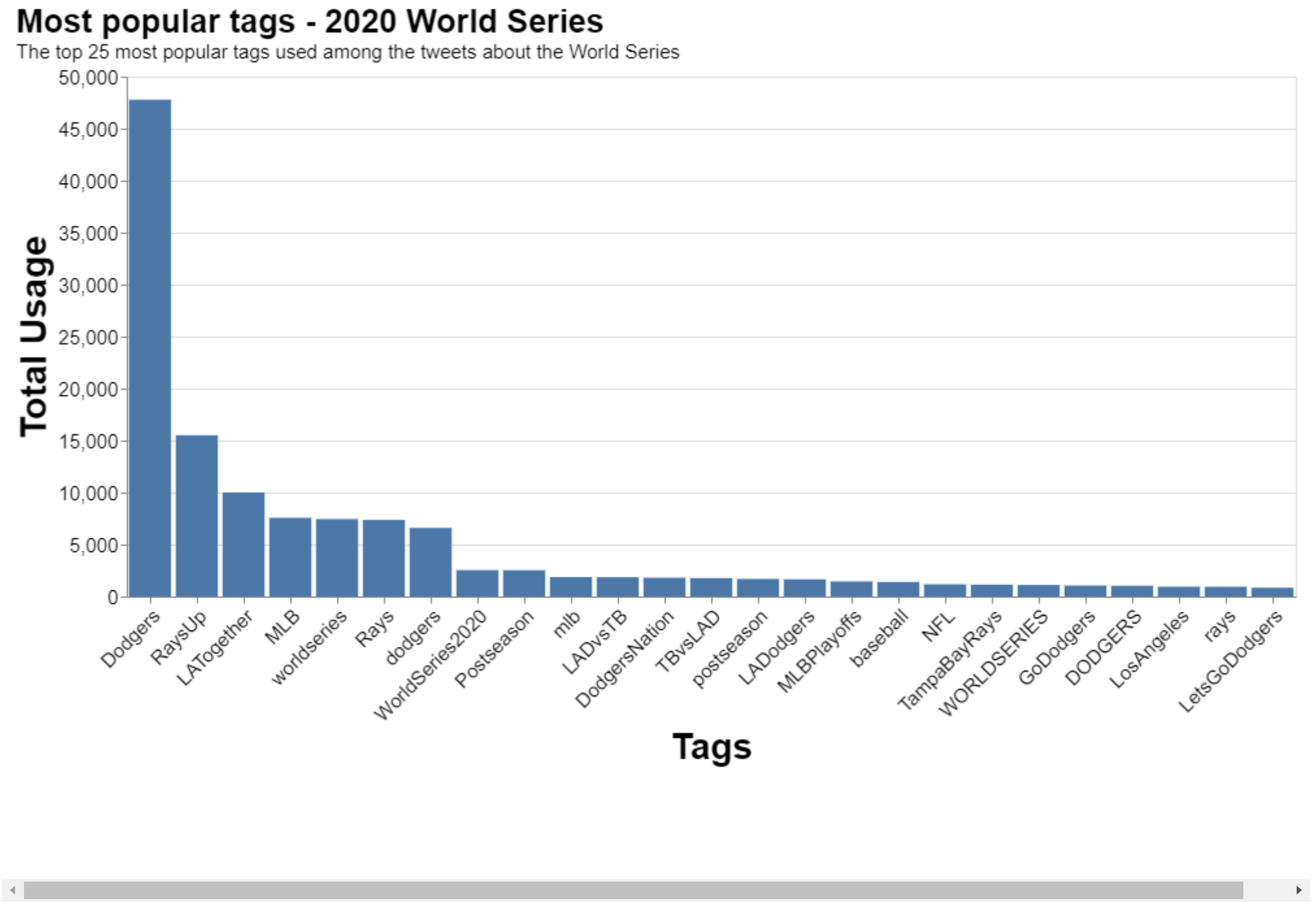
Out[13]:

	tags	count
0	WorldSeries	180391
1	Dodgers	47788
2	RaysUp	15511
3	LATogether	10006
4	MLB	7570
5	worldseries	7447
6	Rays	7371
7	dodgers	6601
8	WorldSeries2020	2543
9	Postseason	2533
10	mlb	1870
11	LADvsTB	1858
12	DodgersNation	1805
13	TBvsLAD	1767
14	postseason	1691
15	LADodgers	1654
16	MLBPlayoffs	1453
17	baseball	1394
18	NFL	1177
19	TampaBayRays	1143
20	WORLDSERIES	1112
21	GoDodgers	1053
22	DODGERS	1032
23	LosAngeles	945
24	rays	941

	tags	count
25	LetsGoDodgers	843
26	COVID19	780
27	ITFDB	721
28	GamblingTwitter	712
29	Worldseries	699
30	raysup	693
31	LA	678
32	NLCS	641
33	Baseball	553
34	Raysup	528
35	BeatLA	523
36	LosAngelesDodgers	521
37	MLBPostseason	511
38	TheBachelorette	503
39	Champions	498
40	WorldSeriesChamps	497
41	NBAFinals	488
42	TampaBay	446
43	Sweepstakes	445
44	SerieMundial	436
45	Game6	429
46	DodgersWin	422
47	GoRays	405
48	Braves	395
49	RallyBuds	381

In [14]:
```
c1=c[1:26]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('tags',sort=['count'],title='Tags',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular tags - 2020 World Series"],
      "subtitle":["The top 25 most popular tags used among the tweets about the World Series"]
    }).configure_axis(
    labelFontSize=15,
    titleFontSize=28
).configure_title(
    anchor='start',
    fontSize = 25,
    subtitleFontSize = 15
).configure_legend(
    titleFontSize=20,
    labelFontSize=16
)
```

Out[14]:


Most popular tags - 2020 World Series
The top 25 most popular tags used among the tweets about the World Series
Total Usage
50,000
45,000
40,000
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
Dodgers
RaysUp
LATogether
MLB
worldseries
Rays
dodgers
WorldSeries2020
Postseason
mlb
LADvsTB
DodgersNation
TBvsLAD
postseason
LADodgers
MLBPlayoffs
baseball
NFL
TampaBayRays
WORLDSERIES
GoDodgers
DODGERS
LosAngeles
rays
LetsGoDodgers
Tags

Emoji

In [15]:
```
count= top_item(df,'emojis')
count.head(20)
```

Out[15]:

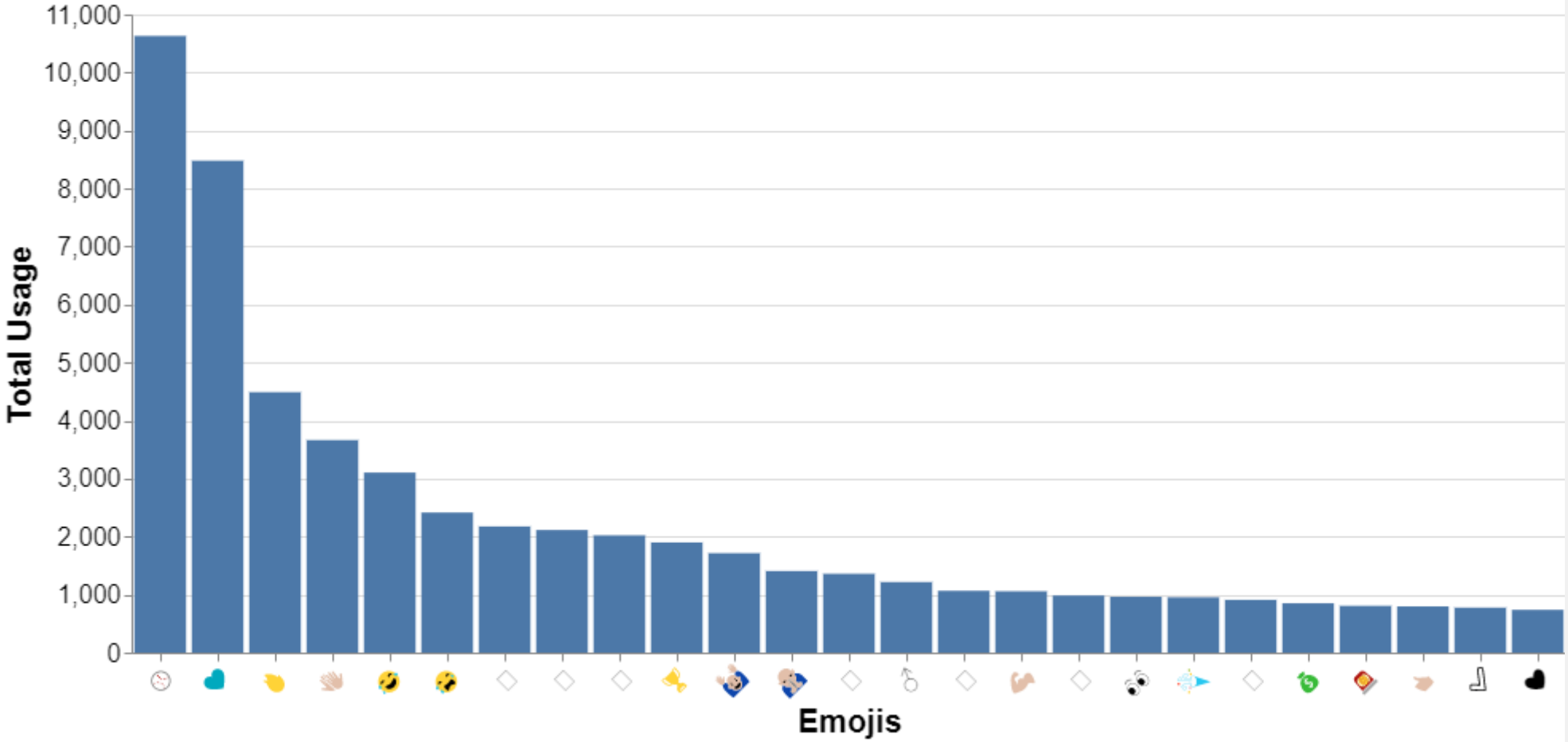
	emojis	count
0	😂	10631
1	❤️	8480
2	🔥	4490
3	🙌	3664
4	😊	3104
5	😍	2415
6		2176
7		2114
8		2022
9	🏆	1900
10	🐻	1715
11	🏂	1407
12		1360
13	♂️	1217
14		1066
15	💪	1058
16		985
17	👀	963
18	🎉	953
19		910

In [16]:
```
c1=count[:25]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('emojis',sort=['count'],title='Emojis',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular emojis - 2020 World Series"],
      "subtitle":["The top 25 most popular emojis used among the tweets about World Series 2020"]
    }).configure_axis(
    labelFontSize=16,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 24,
    subtitleFontSize = 15
)
```

Out[16]:

Most popular emojis - 2020 World Series

The top 25 most popular emojis used among the tweets about World Series 2020



In [17]:
```
c= top_item(df,'words')
c
```

Out[17]:

	words	count
0	game	36499
1	dodgers	23261
2	series	16099
3	win	13913
4	world	13283
5	rays	13201
6	go	12865
7	lets	10826
8	one	9038
9	tonight	8784
10	baseball	8584
11	get	8143
12	snell	7597
13	like	7473
14	im	6834
15	time	6718
16	amp	6347
17	got	5602
18	cash	5588
19	first	5532
20	kershaw	5519
21	mookie	5402
22	team	5348
23	la	5336
24	good	5300
25	going	5228

	words	count
26	last	5020
27	right	4646
28	dont	4622
29	th	4490
30	see	4381
31	back	4374
32	great	4333
33	home	4262
34	year	4248
35	love	4191
36	kevin	4176
37	congrats	3985
38	fans	3959
39	tampa	3870
40	best	3817
41	run	3722
42	night	3692
43	take	3684
44	cant	3673
45	play	3635
46	betts	3601
47	years	3591
48	still	3459
49	inning	3414

Target specific time period for analysis: Game6 (Dodger won championship)

In [18]:
```
df['Date'] = pd.to_datetime(df['date'])
mask = (df['Date'] > '2020-10-27 23:00') & (df['Date'] < '2020-10-28 06:00')
game6 = df.loc[mask].sort_values('Date')
game6 = game6.reset_index()
game6.drop(columns=['index','Date'],inplace=True)

game6
```

Out[18]:

	id	date	text	tags	emojis	clean_text	words	sentiment_score	day	hour	10
0	1321225165804482561	2020-10-27 23:00:01+00:00	The #Rays have Snell going in Game 6, how does...	[Rays, WorldSeries]	[]	the have snell going in game how does that a...	[snell, going, game, affect, thoughts, wins, f...	0.083333	2020-10-27	23	
1	1321225165007630338	2020-10-27 23:00:01+00:00	In Gonsolin We Trust! \n\n#GonsolinDay #WorldS...	[GonsolinDay, WorldSeries, Postseason, BringIt...	[]	in gonsolin we trust	[gonsolin, trust]	0.000000	2020-10-27	23	
2	1321225166563803145	2020-10-27 23:00:01+00:00	The Dodgers are one win away from breaking the...	[WorldSeries]	[]	the dodgers are one win away from breaking the...	[dodgers, one, win, away, breaking, droughtthe...	0.800000	2020-10-27	23	
3	1321225206309027845	2020-10-27 23:00:11+00:00	It's GAME 6 of the World Series TONIGHT at 8...	[DoYouGambetDC, WorldSeries, Game6]	[⚾, 💻,]	its game of the world series tonight at pm	[game, world, series, tonight, pm, series,	0.200000	2020-10-27	23	

In [19]:
```
# compute the 15-minute rolling average percentage of each 5 sentiment degree

senti = game6.groupby(['day','hour','min']).sum()[['POS', 'pos', 'neu','neg', 'NEG']]
senti = senti.reset_index()
senti['date'] = senti['day'] + ' ' + senti['hour'] + ':' + senti['min']
senti['size'] = pd.Series(game6.groupby(['day','hour','min']).size().values)
senti[['POS', 'pos', 'neu', 'neg', 'NEG','15m_count']] = senti.rolling(window=15,min_periods=1).sum()[['POS', 'pos', 'neu', 'neg',
senti['POSITIVE'] = senti['POS'] / senti['15m_count']
senti['positive'] = senti['pos'] / senti['15m_count']
senti['neutral'] = senti['neu'] / senti['15m_count']
senti['negative'] = senti['neg'] / senti['15m_count']
senti['NEGATIVE'] = senti['NEG'] / senti['15m_count']

senti.head()
```

Out[19]:

	day	hour	min	POS	pos	neu	neg	NEG	date	size	15m_count	POSITIVE	positive	neutral	negative	NEGATIVE
0	2020-10-27	23	00	1.0	2.0	9.0	1.0	0.0	2020-10-27 23:00	13	13.0	0.076923	0.153846	0.692308	0.076923	0.0
1	2020-10-27	23	01	1.0	5.0	15.0	2.0	0.0	2020-10-27 23:01	10	23.0	0.043478	0.217391	0.652174	0.086957	0.0
2	2020-10-27	23	02	1.0	8.0	22.0	3.0	0.0	2020-10-27 23:02	11	34.0	0.029412	0.235294	0.647059	0.088235	0.0
3	2020-10-27	23	03	1.0	12.0	34.0	4.0	0.0	2020-10-27 23:03	17	51.0	0.019608	0.235294	0.666667	0.078431	0.0
4	2020-10-27	23	04	1.0	16.0	41.0	5.0	0.0	2020-10-27 23:04	12	63.0	0.015873	0.253968	0.650794	0.079365	0.0

In [20]:
```
# dataframe for visualization

date = []
value = []
label = []

senti_flow = pd.DataFrame()

for i in ['POSITIVE', 'positive', 'neutral', 'negative', 'NEGATIVE']:
    lst=[]
    lst1=[]
    lst2=list(senti.date.values)

    for j in range(len(senti)):
        lst.append(i)
        lst1.append(senti[i][j])

    date += lst2
    value += lst1
    label += lst

senti_flow['date'] = pd.Series(date)
senti_flow['Sentiment_label'] = pd.Series(label)
senti_flow['perct'] = pd.Series(value)
senti_flow.head()
```

Out[20]:

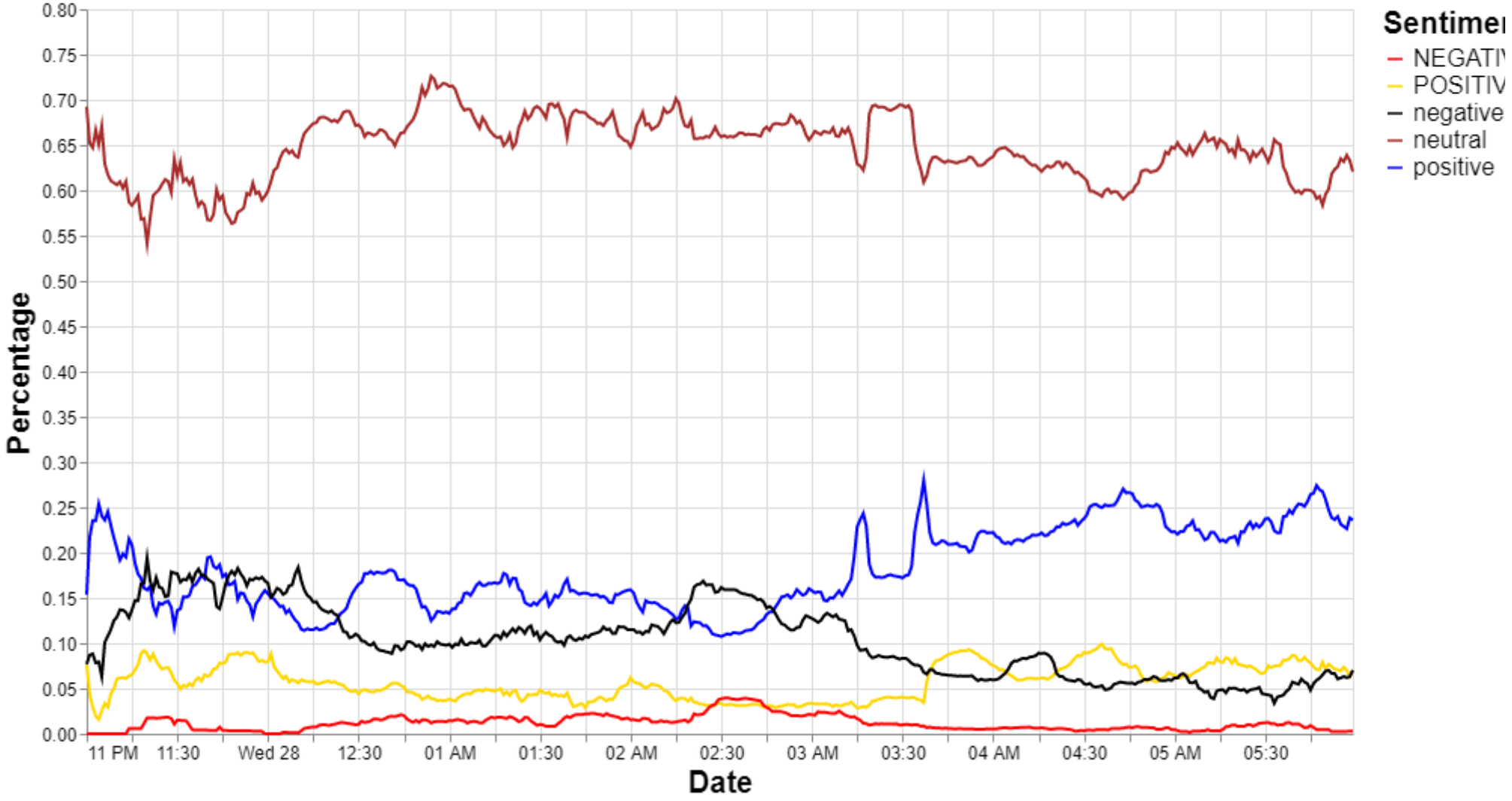
	date	Sentiment_label	perct
0	2020-10-27 23:00	POSITIVE	0.076923
1	2020-10-27 23:01	POSITIVE	0.043478
2	2020-10-27 23:02	POSITIVE	0.029412
3	2020-10-27 23:03	POSITIVE	0.019608
4	2020-10-27 23:04	POSITIVE	0.015873

In [21]:
```
alt.Chart(senti_flow).mark_line().encode(
    x=alt.X('date:T',title='Date'),
    y=alt.Y('perct:Q',title='Percentage'),
    color=alt.Color('Sentiment_label',
                    scale=alt.Scale(
            range=['red', 'gold','black','brown','blue']))
).properties(width=840,height=480,title={
      "text": ["Sentiment Flow - 2020 WorldSeries G6"],
      "subtitle": ["The percentage flow of each degree of sentiment on Twitter, values are computed by 15-minute rolling average"]
    }).configure_axis(
    labelFontSize=12,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 24,
    subtitleFontSize = 15
).configure_legend(
    titleFontSize=20,
    labelFontSize=16
)
```

Out[21]:

Sentiment Flow - 2020 WorldSeries G6

The percentage flow of each degree of sentiment on Twitter, values are computed by 15-minute rolling average



In [22]:
```
c=top_item(game6,'tags')
c
```

Out[22]:

	tags	count
0	WorldSeries	59685
1	Dodgers	15682
2	LATogether	3197
3	dodgers	3136
4	RaysUp	2926
5	worldseries	1846
6	Rays	1732
7	MLB	1539
8	DodgersNation	1484
9	WorldSeries2020	1140
10	TBvsLAD	754
11	LADodgers	704
12	DODGERS	474
13	LosAngeles	439
14	mlb	414
15	GoDodgers	414
16	postseason	370
17	Champions	355
18	Postseason	353
19	COVID19	326
20	WORLDSERIES	321
21	Game6	304
22	rays	260
23	LA	256
24	LakeShow	244
25	NBAFinals	244

	tags	count
26	baseball	230
27	LetsGoDodgers	229
28	MLBPlayoffs	225
29	dodgerswin	219
30	Snell	218
31	ITFDB	207
32	TampaBayRays	207
33	Lakers	199
34	Champs	192
35	2020	169
36	BlakeSnell	169
37	raysup	167
38	champions	165
39	SerieMundial	162
40	MVP	154
41	LADvsTB	151
42	KevinCash	149
43	WorldSeriesChamps	149
44	Worldseries	148
45	Mlb	139
46	champs	139
47	Vs	133
48	mookiebetts	127
49	LosAngelesDodgers	119

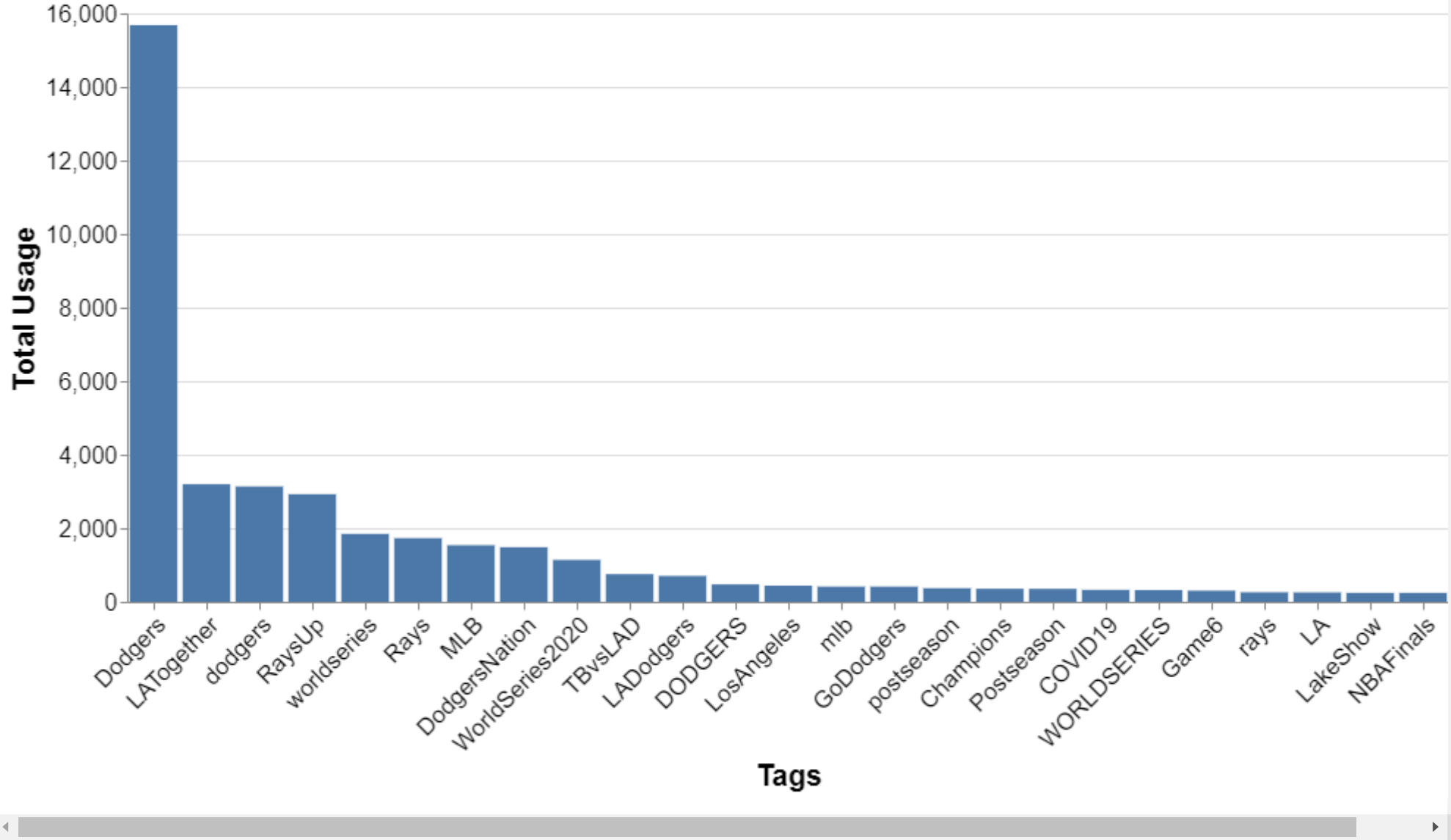
In [23]:

```
c1=c[1:26]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('tags',sort=['count'],title='Tags',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular tags - 2020 WorldSeries G6"],
      "subtitle":["The top 25 most popular tags used among the tweets about WS G6"]
    }).configure_axis(
    labelFontSize=16,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 24,
    subtitleFontSize = 15
)
```

Out[23]:

Most popular tags - 2020 WorldSeries G6

The top 25 most popular tags used among the tweets about WS G6



In [24]:
```
c=top_item(game6,'emojis')
c
```

Out[24]:

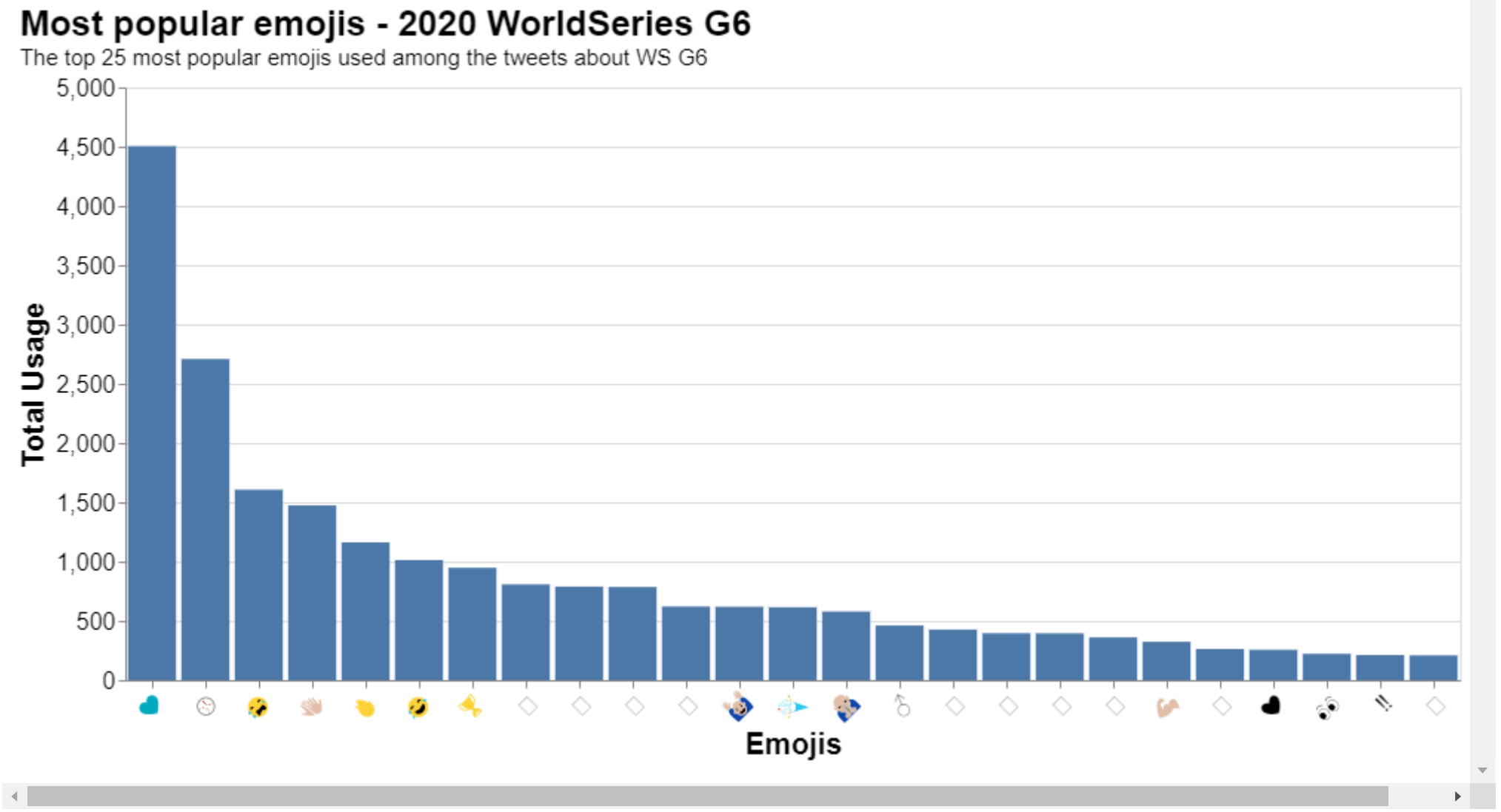
	emojis	count
0	💙	4505
1	😘	2709
2	😂	1607
3	🙌	1474
4	🔥	1162
5	😂	1013
6	🏆	948
7		808
8		789
9		786
10		621
11	🐻	619
12	🎉	615
13	🤦	579
14	♂	461
15		427
16		396
17		395
18		361
19	💪	324
20		263
21	❤	256
22	👀	222
23	‼	212
24		209

	emojis	count
25	💥	199
26		167
27	💯	163
28		160
29	😊	152
30	🤣	143
31	💰	131
32	😩	131
33		129
34	👍	128
35	🤟	127
36	♀	126
37	😟	123
38		117
39	🎉	114
40	✨	114
41		114
42		113
43	🤎	111
44	😬	111
45	✓	107
46	🌴	106
47	😍	98
48		93
49	😕	86

In [25]:

```
c1=c[:25]

alt.Chart(c1).mark_bar().encode(
    x=alt.X('emojis',sort=['count'],title='Emojis',axis=alt.Axis(labelAngle=-45)),
    y=alt.Y('count',title='Total Usage')
).properties(width=900,height=400,title={
      "text": ["Most popular emojis - 2020 WorldSeries G6"],
      "subtitle":["The top 25 most popular emojis used among the tweets about WS G6"]
    }).configure_axis(
    labelFontSize=16,
    titleFontSize=20
).configure_title(
    anchor='start',
    fontSize = 24,
    subtitleFontSize = 15
)
```

Out[25]:


Most popular emojis - 2020 WorldSeries G6
The top 25 most popular emojis used among the tweets about WS G6
Total Usage
5,000
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
Emojis

In [ ]: