Spanish Tweet Classifier

Business Context. Transfer learning has revolutionized image analysis, because now we have Convolutional Neural Networks (CNNs) pre-trained for hundreds of expensive hours on tens of thousands of varied images. A CNN that already knows how to see requires very little training data to re-train for a specific task in machine vision.

The same concept applies to the Recurrent Neural Networks (RNNs) used for Natural Language Processing (NLP). It takes a lot of expensive computation (and lots of data) to train an RNN to understand a given language. However, that same language model can be re-trained much more easily to understand a particular kind of text written in that language, such as tweets.

Business Problem. Many social networks analyze the communications that run through them, and would be interested in classifying them according to whether they exhibit positive or negative sentiment.

Analytical Context. The same RNN pre-trained on Spanish can be trained on medical texts, legal texts, customer complaints, poetry, novels, movie scripts, or almost anything else. We will re-train an existing **recurrent neural network (RNN)** to deal with the nuances of Spanish tweet language. We will then use this to classify a set of Spanish tweets. Afterwards, we will interpret the results of our classifier.

Notebook setup (10 mts)

Setup for Colab

If you have this notebook in your Google Drive, you can open it with Google Colab. Run the following cells to mount your Drive and make its files accessible from within the notebook. You will be prompted to open a URL and retrieve a code, which you'll paste in the field below:

In [1]:
```
# This cell will prompt you to connect this notebook with your google a
ccount.

from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
root_dir = "/content/gdrive/My Drive/"
base_dir = root_dir + 'Colab Notebooks/case_17.3'
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?
client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuser
content.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_t
ype=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.t
est%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fw
ww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.go
ogleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/gdrive
```

In [2]:
```
# Set up Colab to have the ideal settings for fast.ai
!curl -s https://course.fast.ai/setup/colab | bash
```

```
Updating fastai...
Done.
```

Fast ai

Setup for Jupyter Notebooks

Run this cell if operating locally or on AWS as a Jupyter notebook:

In [1]:
```
import os
base_dir = os.getcwd()
```

Setup for everyone

Run these cells from wherever you are operating (Colab or Jupyter):

In [2]:
```python
import pandas as pd
import numpy as np
import sklearn
from fastai.text import *
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

In [3]:
```python
data_path = Path(base_dir + '/tweets')
model_path = Path(base_dir + '/models')
data_path
```

Out[3]:
```
PosixPath('/Users/haris.jaliawala/Downloads/case_17.3/tweets')
```

Preparing the data (20 mts)

In this case, we'll be using a corpus of Spanish language tweets with about 250,000 tweets. They are crudely classified as either positive or negative based on whether the tweet contained one of the following simple emojis:

`:)` or `:-)` = positive

`:(` or `:-(` = negative

It's a relatively crude way of determining sentiment, but it works. We'll clean those tweets to remove non-ASCII characters as well as the instances of these emojis. We have three pre-classified lists of tweets: positive, negative, and neutral (anything which didn't contain one of the above emojis):

In [4]:
```python
pos = pd.read_table(data_path/'tweets_pos_clean.txt', header=None)
neg = pd.read_table(data_path/'tweets_neg_clean.txt', header=None)
neut = pd.read_table(data_path/'tweets_clean.txt', header=None)
```

In [5]:
```python
# Verify the size of the dataframes
pos.shape, neg.shape, neut.shape
```

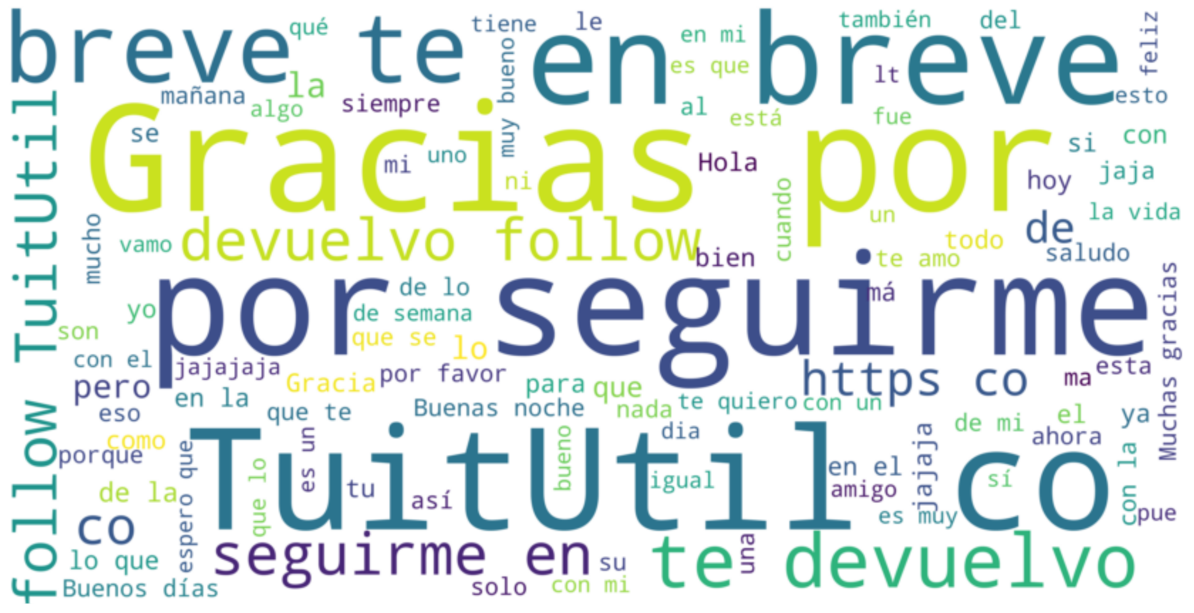
```
Out[5]: ((55056, 1), (120948, 1), (70194, 1))
```

```
In [7]: for tweet in pos.head(10)[0]:
          print(tweet)
```

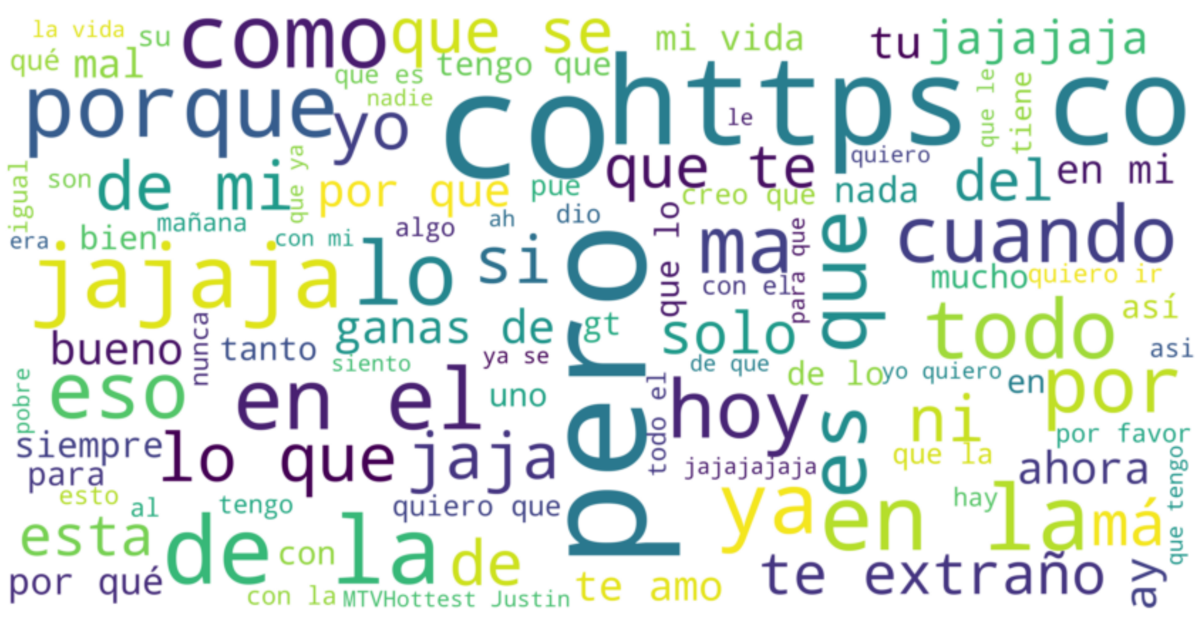
```
Se imaginan a los chicos agradeciendo por el premio con cara de orgull
o?.Que bonito :).#MTVHottest One Direction
Eclesiastes4:9-12 ♡ Siempre, promesa :)  https://t.co/XbrYsqa43T
@pedroj_ramirez Qué saborío, PJ. ya no compartes ni un gintonic con nos
otros. :)
Buenos dias para todos. Feliz inicio de semana. :-) http://t.co/svMgEca
xLr
@pepedom @bquintero Gracias! No es así, deja claro que es el 100% de aq
uí http://t.co/cD3VFu7hnH }:)
Solo 1 :) http://t.co/TNWvSO2Gfa
siempre estuviste a mi lado :)
Somos los más felices con #fox y su #FeriadoSimpson así si es un #domin
gofeliz  :) http://t.co/LQjUC6k3Nq
@POetaVIPGT awww así debe de ser :)
@mundodms :) .Te Ayudo ? ☺
```

We can get further insight by generating separate word clouds for the positive, negative, and neutral tweets.

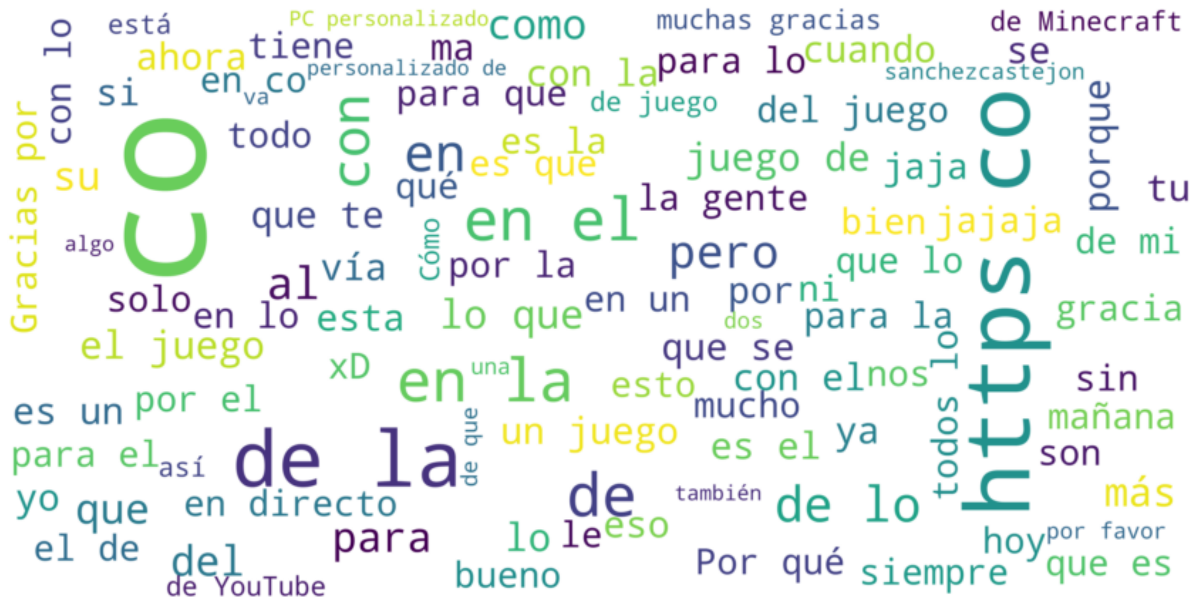
```
In [8]: # Positive
        word_cloud_text = ''.join(pos[0].tolist())
        wordcloud = WordCloud(max_font_size=100, max_words=100, background_colo
        r="white",\
                                   scale = 10,width=800, height=400).generate(wo
        rd_cloud_text)

        plt.figure(figsize = (15, 30))
        plt.imshow(wordcloud, interpolation="bilinear")
        plt.axis("off")
        plt.show()
```



```
In [9]: # Negative
        word_cloud_text = ''.join(neg[0].tolist())
        wordcloud = WordCloud(max_font_size=100, max_words=100, background_colo
        r="white",\
                                      scale = 10,width=800, height=400).generate(wo
        rd_cloud_text)

        plt.figure(figsize = (15, 30))
        plt.imshow(wordcloud, interpolation="bilinear")
        plt.axis("off")
        plt.show()
```



In [10]:
```
# Neutral
word_cloud_text = ''.join(neut[0].tolist())
wordcloud = WordCloud(max_font_size=100, max_words=100, background_colo
r="white",\
                            scale = 10,width=800, height=400).generate(wo
rd_cloud_text)

plt.figure(figsize = (15, 30))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



Exercise 1

What do you notice in these word clouds? How can we clean them up?

Answer

We'll at least clean up the following:

1. The emoji used to classify the tweets in the first place.
2. Non-ASCII characters.
3. http://hyperlinks
4. @user tags
5. #hashtags

In [11]:
```
for tweet in pos.head(10)[0]:
  print(tweet)
```

```
Se imaginan a los chicos agradeciendo por el premio con cara de orgull
o?.Que bonito :).#MTVHottest One Direction
Eclesiastes4:9-12 ♡ Siempre, promesa :)  https://t.co/XbrYsqa43T
@pedroj_ramirez Qué saborío, PJ. ya no compartes ni un gintonic con nos
otros. :)
Buenos dias para todos. Feliz inicio de semana. :-) http://t.co/svMgEca
xLr
@pepedom @bquintero Gracias! No es así, deja claro que es el 100% de aq
uí http://t.co/cD3VFu7hnH }:)
Solo 1 :) http://t.co/TNWvSO2Gfa
siempre estuviste a mi lado :)
Somos los más felices con #fox y su #FeriadoSimpson así si es un #domin
gofeliz  :) http://t.co/LQjUC6k3Nq
@POetaVIPGT awww así debe de ser :)
@mundodms :) .Te Ayudo ? ☺
```

In [12]:
```
def cleanup(text):
    '''
    Removes annoying non-text from our tweet library
    '''
    # Remove the classification emoji
    text = text.replace(':)','')\
            .replace(':-)','')\
            .replace(':(','')\
            .replace(':-(','')\
            .replace(':D','')

    # Removes other emoji
    emoji_pattern = re.compile("["
                u"\U0001F600-\U0001F64F"  # emoticons
                u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                u"\U0001F680-\U0001F6FF"  # transport & map symbols
                u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                u"\U00002702-\U000027B0"
                u"\U000024C2-\U0001F251"
                u"\U0001f926-\U0001f937"
```

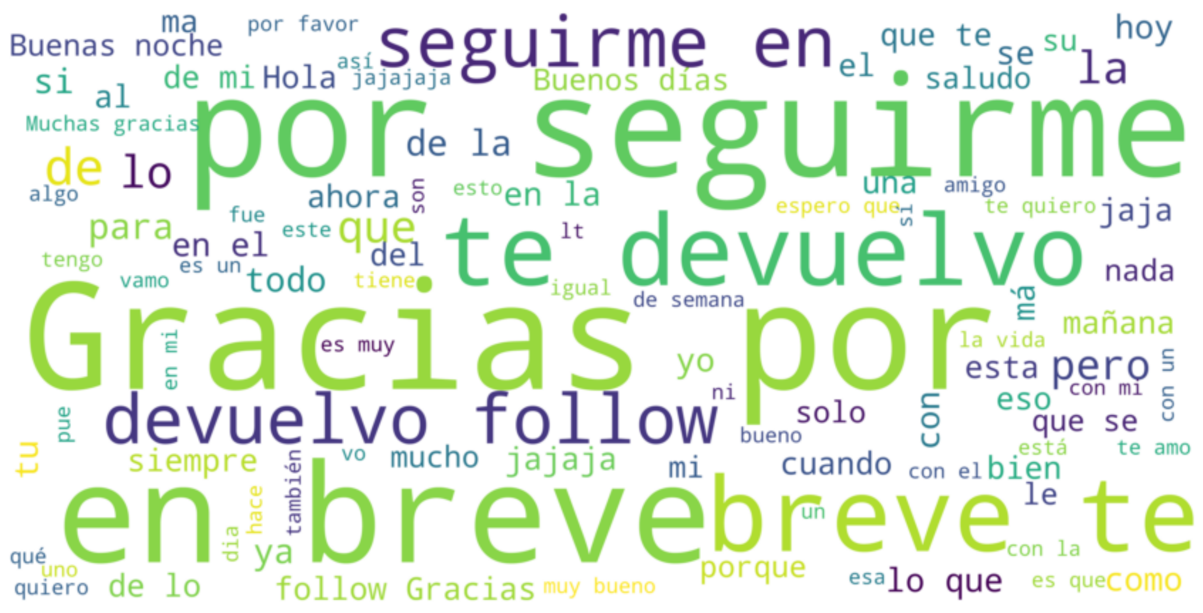
```
                 u'\U00010000-\U0010ffff'
                 u"\u200d"
                 u"\u2640-\u2642"
                 u"\u2600-\u2B55"
                 u"\u23cf"
                 u"\u23e9"
                 u"\u231a"
                 u"\u3030"
                 u"\ufe0f"
    "]+", flags=re.UNICODE)

    text = re.sub(emoji_pattern, '', text)

    # Delete @user
    text = re.sub(r'@[^\s]+','',text)
    # Delete hyperlinks
    text = re.sub(r'https?:\/\/.*\/\w*', '', text)
    # Delete hashtags
    text = re.sub(r'#\w*', '', text)

    return text

for df in [pos, neg, neut]:
  df['tweet'] = df[0].apply(cleanup)
```

In [13]:
```
for i in range(10):
  print(pos[0][i])
  print(pos['tweet'][i])
  print()
```

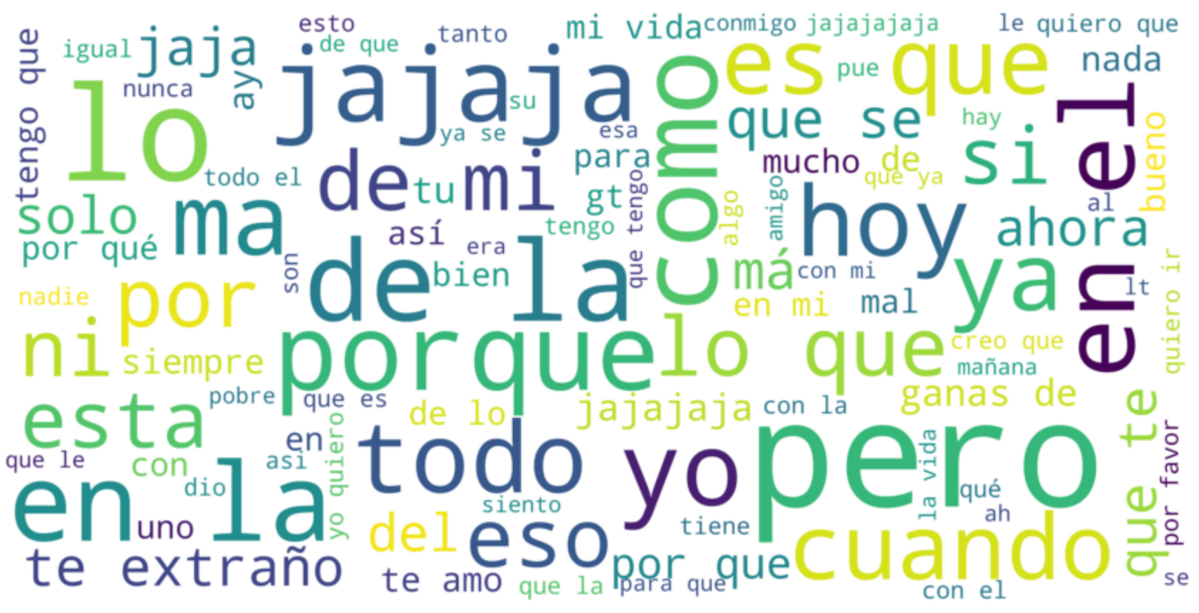
```
Se imaginan a los chicos agradeciendo por el premio con cara de orgull
o?.Que bonito :).#MTVHottest One Direction
Se imaginan a los chicos agradeciendo por el premio con cara de orgull
o?.Que bonito . One Direction

Eclesiastes4:9-12 ♡ Siempre, promesa :)  https://t.co/XbrYsqa43T
Eclesiastes4:9-12  Siempre, promesa

@pedroj_ramirez Qué saborío, PJ. ya no compartes ni un gintonic con nos
```

```
otros. :)
 Qué saborío, PJ. ya no compartes ni un gintonic con nosotros.

Buenos dias para todos. Feliz inicio de semana. :-) http://t.co/svMgEca
xLr
Buenos dias para todos. Feliz inicio de semana.

@pepedom @bquintero Gracias! No es así, deja claro que es el 100% de aq
uí http://t.co/cD3VFu7hnH }:)
  Gracias! No es así, deja claro que es el 100% de aquí  }

Solo 1 :) http://t.co/TNWvSO2Gfa
Solo 1

siempre estuviste a mi lado :)
siempre estuviste a mi lado

Somos los más felices con #fox y su #FeriadoSimpson así si es un #domin
gofeliz  :) http://t.co/LQjUC6k3Nq
Somos los más felices con  y su  así si es un

@POetaVIPGT awww así debe de ser :)
 awww así debe de ser

@mundodms :) .Te Ayudo ? ☺
  .Te Ayudo ?
```

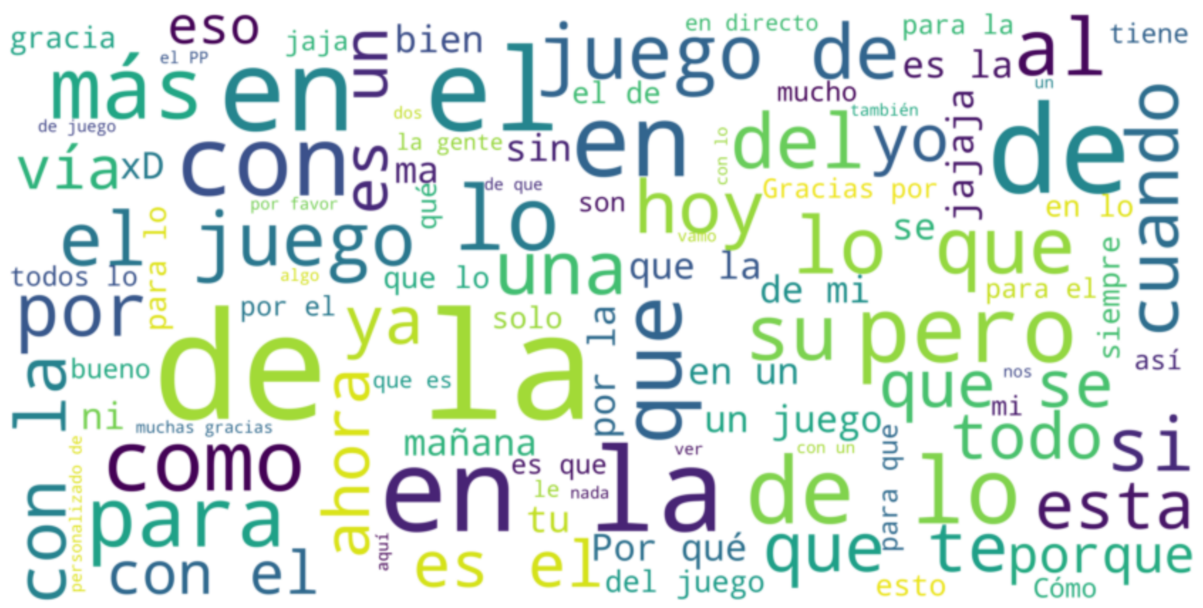
In [14]:
```python
# Positive
word_cloud_text = ''.join(pos['tweet'].tolist())
wordcloud = WordCloud(max_font_size=100, max_words=100, background_colo
r="white",\
                          scale = 10,width=800, height=400).generate(wo
rd_cloud_text)

plt.figure(figsize = (15, 30))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



In [15]:
```
# Negative
word_cloud_text = ''.join(neg['tweet'].tolist())
wordcloud = WordCloud(max_font_size=100, max_words=100, background_colo
r="white",\
                              scale = 10,width=800, height=400).generate(wo
rd_cloud_text)

plt.figure(figsize = (15, 30))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



```
In [16]: # Neutral
         word_cloud_text = ''.join(neut['tweet'].tolist())
         wordcloud = WordCloud(max_font_size=100, max_words=100, background_colo
         r="white",\
                                                 scale = 10,width=800, height=400).generate(wo
         rd_cloud_text)

         plt.figure(figsize = (15, 30))
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis("off")
         plt.show()
```



Now that we have only clean emoji, we can put them in a single dataframe to train the language model. Note how we're using ALL the tweets we have, regardless of whether we know their label, because we want to train the language model on as many Spanish tweets as possible:

In [17]:
```
all_tweets = pd.concat([pos['tweet'],
                        neg['tweet'],
                        neut['tweet']])\
               .reset_index(drop=True)

all_tweets = pd.DataFrame(all_tweets)
all_tweets.head()
```

Out[17]:

	tweet
0	Se imaginan a los chicos agradeciendo por el p...
1	Eclesiastes4:9-12 Siempre, promesa

	tweet
2	Qué saborío, PJ. ya no compartes ni un ginton...
3	Buenos dias para todos. Feliz inicio de semana.
4	Gracias! No es así, deja claro que es el 100...

Let's create the data object now, split into train (90%) and test (10%) sets:

In [18]:
```
# Create data object containing all the tweets,
# which we'll use to train the language model learner.
data_lm = (TextList.from_df(all_tweets, cols='tweet')
        .split_by_rand_pct(0.1, seed=42)    # Split into train and test
        .label_for_lm()                     # Label for language model
        .databunch(bs=128, num_workers=1))  # Turn into a data bunch
```

In [19]:
```
data_lm.save(model_path/'textlist_alltweets')
```

In [20]:
```
data_lm
```

Out[20]:
```
TextLMDataBunch;

Train: LabelList (221579 items)
x: LMTextList
xxbos xxmaj se imaginan a los chicos agradeciendo por el premio con car
a de xxunk bonito . xxmaj one xxmaj direction,xxbos xxmaj xxunk - 12 xx
maj siempre , promesa,xxbos xxmaj qué xxunk , xxup pj . ya no compartes
ni un xxunk con nosotros .,xxbos xxmaj buenos dias para todos . xxmaj f
eliz inicio de semana .,xxbos xxmaj gracias ! xxmaj no es así , deja cl
aro que es el 100 % de aquí }
y: LMLabelList
,,,,
Path: .;

Valid: LabelList (24619 items)
x: LMTextList
```

```
xxbos no tengo , pero me refería a que quiero hacer xxunk contigo , en
el mismo equipo xddd,xxbos a veces extraño que seas mi amiga porque sie
mpre me seguías en mis locuras,xxbos xxmaj yo leo " xxmaj xxunk " y me
acuerdo inmediatamente de,xxbos xxmaj estoy pobre , triste y sola . xxm
aj haay,xxbos y aquí está : xxmaj apple xxmaj watch -
y: LMLabelList
,,,,
Path: .;

Test: None
```

The basic steps of NLP processing are all contained in the previous step. `fastai` has created a vocabulary with all the words that showed up at least a few times in the corpus, and added many symbols of its own to mark other elements of the tweets. For example:

1. `xxbos` marks the beginning of a string
2. `xxunk` replaces unknown words. Most of them are just so rare that they didn't make it into the vocabulary
3. `xxmaj` means that the next word will be uppercase

The tokenization process has also separated out symbols into their own words.

In [21]:
```
data_lm.show_batch()
```

idx	text
0	claro que es el 100 % de aquí } xxbos xxmaj solo 1 xxbos siempre estuviste a mi lado xxbos xxmaj somos los más felices con y su así si es un xxbos awww así debe de ser xxbos .te xxmaj ayudo ? xxbos " xxmaj las abuelas xxup siempre saben . \ m / " xxmaj yeah ! ! } xxbos xxmaj gracias por seguirme , en breve te
1	de xxup vs . xxmaj andre xxmaj agus , xxmaj jose y xxmaj emi xxup ya xxup nos xxup siguen xxbos xxmaj por favor , no porque xxunk harán . y voy a enojarme tienen que decirles que fuera en español)) xxbos xxmaj tu tienes algo que no se que es xxrep 4 . xxmaj intensamente xxbos xxmaj hey xxmaj escucha xxmaj lo xxmaj nuevo " xxmaj me
2	te recomiendo desayunos a domicilio xxbos xxmaj alfin xxmaj empezó xxmaj duro de xxmaj xxunk xxbos xxmaj ya me vale q pase mañana . xxmaj dios me ha enseñado que puedo ser feliz con lo poco .. xxbos xxmaj hola escucha mi nuevo tema " xxmaj maravilloso xxmaj dios " , compartelo , suscribete y dale like : xxbos xxmaj xxunk xxmaj tio xxmaj xxunk . xxbos xxmaj hace bien

idx	text
3	ti por tu talento y tu buena onda en toda la grabación ! ! xxmaj tengo mucho cariño a tu querido xxbos @ xxmaj xxunk , xxmaj santander xxbos xxmaj foto 2012 :'(te extraño tanto mi ángel xxmaj xxunk y xxunk desde el cielo xxup xxunk xxmaj te amooo ! ! ! 14 / 11 / 2013 xxbos xxmaj esperó que me valla bien y si es así me
4	que tal xxbos dios quiera que no slds buenas noches yami xxbos xxmaj ajajajajaj ok xxbos 1er episodio .. xxmaj jajaj eh ... xxbos xxmaj gracias por seguirme , en breve te devuelvo follow xxbos soy un niño incomprendido xxrep 4) xxbos yo no puedo con los xxmaj iphones , nunca me han gustado y creo q nunca me gustaran . xxmaj pero que bueno que te sacaste el

In addition to our dataframe and data bunch of all the tweets, we want a second set with only the labeled tweets, which we'll use to train the classifier:

In [22]:
```
# Create a dataframe of labeled tweets
pos['mood'] = 'positive'
neg['mood'] = 'negative'

labeled_tweets = pd.concat([pos[['tweet','mood']], neg[['tweet','mood'
]]])\
                    .reset_index(drop=True)

labeled_tweets.sample(5)
```

Out[22]:

	tweet	mood
72925	Tengo un grano de tras de la Nariz	negative
117171	Que mala que es conmigo jaja	negative
144290	Yo no tengo friendzoneado a ese bobo hijueputa...	negative
174571	Ketarde!.Ojalá hubiera un camión que pasé aquí...	negative
92298	ah pero estoy aburrido perdón	negative

Let's created a data object containing all the labeled tweets, which we'll feed the classifier learner. Note that we want it to have the full vocabulary of the `data_lm` data bunch, which was trained on more tweets and includes more words. Here we are also automatically generating a validation set of 10%:

In [23]:
```
# Create data object containing all the labeled tweets
data_clas = (TextList.from_df(df=labeled_tweets,
                              path='.',
                              vocab=data_lm.vocab,
                              cols='tweet')
                    .split_by_rand_pct(0.1, seed=42)
                    .label_from_df(cols='mood')
                    .databunch(bs=128, num_workers=1))
```

In [24]:
```
data_clas.train_ds
```

Out[24]:
```
LabelList (158404 items)
x: TextList
xxbos xxmaj se imaginan a los chicos agradeciendo por el premio con car
a de xxunk bonito . xxmaj one xxmaj direction,xxbos xxmaj xxunk - 12 xx
maj siempre , promesa,xxbos xxmaj qué xxunk , xxup pj . ya no compartes
ni un xxunk con nosotros .,xxbos xxmaj buenos dias para todos . xxmaj f
eliz inicio de semana .,xxbos xxmaj gracias ! xxmaj no es así , deja cl
aro que es el 100 % de aquí }
y: CategoryList
positive,positive,positive,positive,positive
Path: .
```

In [25]:
```
data_clas.show_batch()
```

text	target
xxbos xxmaj retweet xxmaj si extrañas las notas altas de xxmaj zayn " pues claro que sí xxmaj one xxmaj direction \n severo incendio \n xxmaj yo debería andar con mis botas y en ese baile . xxmaj pero no , aquí estoy haciendo tarea ! ! ! \n a mi me faltó la 53 \n xxmaj kranevitter todavia no se fue y ya lo extraño	negative
xxbos xxup ya xxup me xxup desmotive xxup ni xxup ganas xxup de xxup ir a xxup jugar xxup el xxup futbol xxup el xxup martes xxup me xxup siento xxup cn xxup baja xxup autoestima xxup necesito xxup estar xxup motivado para xxup jugar xxup si xxup no xxup no xxup puedo	negative

text	target
xxbos — ¡ xxmaj cancelar ! .. — ¡ ¡ xxmaj cancelar ! ! .. — ¡ ¡ ¡ xxmaj xxunk ! ! ! .. — ¡ ¡ xxmaj xxunk ! ! .. — ¡ xxup xxunk ! .. — xxmaj tu mensaje ha sido enviado con éxito . .. — xxmaj mierda	negative
xxbos xxup voy a xxup estar xxup en xxup el xxup cole xxup xxunk xxup del xxup cole xxup oo xxup me xxup voy xxup al xxup baño xxup en xxup el xxup recreo y xxup .todo xxup por xxup vos xxup and xxrep 4 i xxup ayudo xxup xxunk	negative
xxbos xxmaj el : xxmaj estas xxunk .. xxmaj ella : si ? .. xxmaj el : xxmaj bueno me voy ? ! ! .. xxmaj ella : a donde te vas ? .. xxmaj el : a buscar xxmaj trabajo y a dar lo mejor para ustedes ?	positive

In [26]:
```
data_clas.save(model_path/'textclassdatabunch_labeled_tweets')
```

Training the language model (20 mts)

We could train an RNN from scratch, exposing it to a large corpus of Spanish language until it learned how the language works. For example, this video from `fastai` shows how to download all of Spanish language Wikipedia in order to train language models.

However, this process requires a long time (~10 hours), even on really fast computers. The point of transfer learning is to *not* do this from scratch every time. Instead, we want to take a language model that has already been trained on Spanish text, and re-train it only on our tweets for our purposes.

Here is a language model that has already been trained on Spanish text. It uses the AWD_LSTM architecture.

Now that we have our data in the proper objects, let's re-train our language model to understand the very specific kind of Spanish that happens in tweets:

In [27]:
```
# The pre-trained model we have had a slightly different structure than
 the latest
# architecture in fastai.  We need the downloaded model below to have t
he same
# number of hidden layers that were used for training.
config = awd_lstm_lm_config.copy()
```

```
config['n_hid'] = 1150

# Pre-trained language model parts
lm_fns = (model_path/'model-30k-vocab-noqrnn', # Model weights
          model_path/'itos_pretrained') # Match between vocab and token
s

# Create language model learner
learn_lm = language_model_learner(data=data_lm,
                                  arch=AWD_LSTM, # Architecture
                                  config=config,
                                  pretrained_fnames=lm_fns,
                                  drop_mult=0.3)   # multiplier for all
 dropout parameters
```

In [28]:
```
# Parameters for training
bs = 128      # batch size
lr = 1e-3     # learning rate
lr *= bs/48
```

As before, we'll train using the 1cycle policy. The other parameters (learning rates, moments, etc.) have been empirically shown to work for similar NLP tasks and represent good default values:

In [ ]:
```
learn_lm.fit_one_cycle(2, lr*10, moms=(0.8,0.7))
```

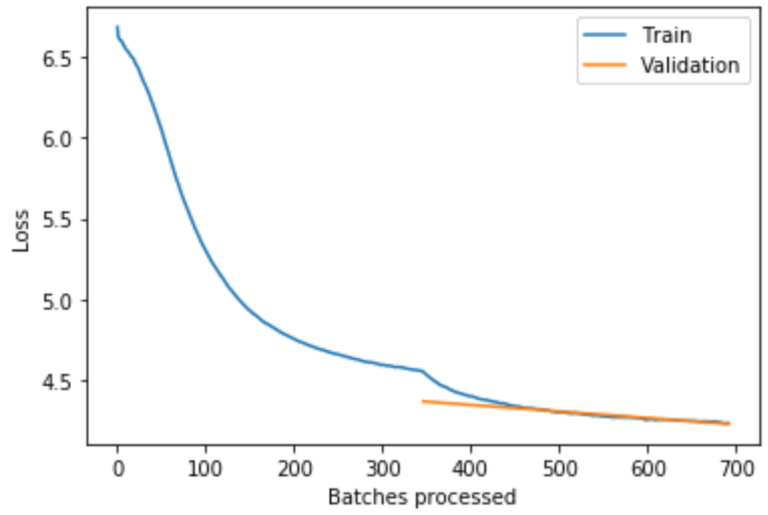
```
0.00% [0/2 00:00<00:00]
```

epoch	train_loss	valid_loss	accuracy	time

```
92.77% [321/346 18:04:50<1:24:29 4.5768]
```

In [0]:
```
learn_lm.save(model_path/'tweet_model')
learn_lm.save_encoder(model_path/'tweet_encoder')
```

In [90]:
```
learn_lm.recorder.plot_losses()
```

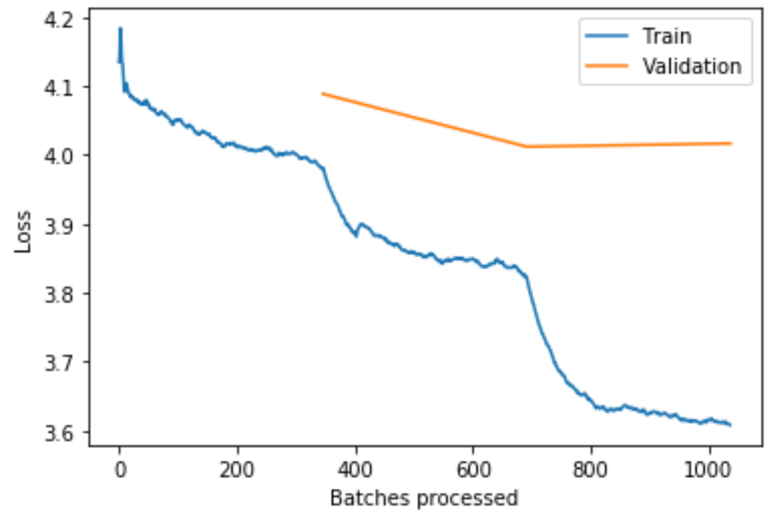


```
In [0]: learn_lm.load(model_path/'tweet_model');
```

```
In [92]: learn_lm.unfreeze()
         learn_lm.fit_one_cycle(3, lr, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	3.979298	4.088369	0.296142	01:59
1	3.823946	4.011726	0.307372	01:58
2	3.607996	4.016504	0.309034	01:59

```
In [93]: learn_lm.recorder.plot_losses()
```



In [0]:
```
learn_lm.save(model_path/'tweet_model_2')
learn_lm.save_encoder(model_path/'tweet_encoder_2')
```

We stop training once the validation score stops improving. At this point, the accuracy is almost 40%, meaning that the model can predict the next word about 40% of the time. To demonstrate this, we can directly feed the model a starting point and see what it predicts will come next:

In [105]:
```
test_text = 'Hola, muchachos, que piensan sobre'
learn_lm.predict(test_text, n_words=30)
```

Out[105]:
```
'Hola, muchachos, que piensan sobre mi abuela , ¿ es tú única salida ?
xxbos Si alguien ser Informático me gusta Mi Tumblr Así … xxbos ¡ Esos'
```

Training a tweet classifier (20 mts)

We started out with a language learner that understood Spanish, and trained it on 250k tweets to understand the rare dialect that is Twitter Spanish. Now we can create a new language model that will learn to classify tweets according to their mood:

```
In [0]:  # As before, we need to adjust the number of hidden layers
         config = awd_lstm_clas_config
         config['n_hid'] = 1150

         learn_clas = text_classifier_learner(data = data_clas, # Classification
          data, labeled
                                              arch = AWD_LSTM,
                                              drop_mult=0.5,
                                              config=config).to_fp16();

         # Use the language encoder that we generated when training on tweets
         learn_clas.load_encoder(model_path/'tweet_encoder_2');
```

```
In [107]: learn_clas
```

```
Out[107]: RNNLearner(data=TextClasDataBunch;

          Train: LabelList (158404 items)
          x: TextList
          xxbos xxmaj se imaginan a los chicos agradeciendo por el premio con car
          a de xxunk bonito . xxmaj one xxmaj direction,xxbos xxmaj xxunk - 12 xx
          maj siempre , promesa,xxbos xxmaj qué xxunk , xxup pj . ya no compartes
          ni un xxunk con nosotros .,xxbos xxmaj buenos dias para todos . xxmaj f
          eliz inicio de semana .,xxbos xxmaj gracias ! xxmaj no es así , deja cl
          aro que es el 100 % de aquí }
          y: CategoryList
          positive,positive,positive,positive,positive
          Path: .;

          Valid: LabelList (17600 items)
          x: TextList
          xxbos xxmaj lo extraño tanto ( (,xxbos xxmaj buenos días hoy toca ^_^,x
          xbos xxmaj mi estado de animo ahora mismo es una mierda . xxmaj por sue
          rte tengo a unas amigas que me han reír con sus estupideces y eso me ay
          uda .,xxbos xxmaj necesito urgentemente que alguien me xxunk , no se qu
          é hacer ..,xxbos xxmaj no me abandonen
          y: CategoryList
          negative,positive,positive,negative,negative
          Path: .;
```

```
Test: None, model=SequentialRNN(
  (0): MultiBatchEncoder(
    (module): AWD_LSTM(
      (encoder): Embedding(32256, 400, padding_idx=1)
      (encoder_dp): EmbeddingDropout(
        (emb): Embedding(32256, 400, padding_idx=1)
      )
      (rnns): ModuleList(
        (0): WeightDropout(
          (module): LSTM(400, 1150, batch_first=True)
        )
        (1): WeightDropout(
          (module): LSTM(1150, 1150, batch_first=True)
        )
        (2): WeightDropout(
          (module): LSTM(1150, 400, batch_first=True)
        )
      )
      (input_dp): RNNDropout()
      (hidden_dps): ModuleList(
        (0): RNNDropout()
        (1): RNNDropout()
        (2): RNNDropout()
      )
    )
  )
  (1): PoolingLinearClassifier(
    (layers): Sequential(
      (0): BatchNorm1d(1200, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
      (1): Dropout(p=0.2, inplace=False)
      (2): Linear(in_features=1200, out_features=50, bias=True)
      (3): ReLU(inplace=True)
      (4): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_
running_stats=True)
      (5): Dropout(p=0.1, inplace=False)
      (6): Linear(in_features=50, out_features=2, bias=True)
    )
  )
), opt_func=functools.partial(<class 'torch.optim.adam.Adam'>, betas=
(0.9, 0.99)), loss_func=FlattenedLoss of CrossEntropyLoss(), metrics=[<
function accuracy at 0x7f471f0168c8>], true_wd=True, bn_wd=True, wd=0.0
1, train_bn=True, path=PosixPath('.'), model_dir='models', callback_fns
=[functools.partial(<class 'fastai.basic_train.Recorder'>, add_time=Tru
e, silent=False)], callbacks=[RNNTrainer
learn: RNNLearner(data=TextClasDataBunch;

Train: LabelList (158404 items)
x: TextList
xxbos xxmaj se imaginan a los chicos agradeciendo por el premio con car
a de xxunk bonito . xxmaj one xxmaj direction,xxbos xxmaj xxunk - 12 xx
maj siempre , promesa,xxbos xxmaj qué xxunk , xxup pj . ya no compartes
ni un xxunk con nosotros .,xxbos xxmaj buenos dias para todos . xxmaj f
eliz inicio de semana .,xxbos xxmaj gracias ! xxmaj no es así , deja cl
aro que es el 100 % de aquí }
y: CategoryList
positive,positive,positive,positive,positive
Path: .;

Valid: LabelList (17600 items)
x: TextList
xxbos xxmaj lo extraño tanto ( (,xxbos xxmaj buenos días hoy toca ^_^,x
xbos xxmaj mi estado de animo ahora mismo es una mierda . xxmaj por sue
rte tengo a unas amigas que me han reír con sus estupideces y eso me ay
uda .,xxbos xxmaj necesito urgentemente que alguien me xxunk , no se qu
é hacer ..,xxbos xxmaj no me abandonen
y: CategoryList
negative,positive,positive,negative,negative
Path: .;

Test: None, model=SequentialRNN(
  (0): MultiBatchEncoder(
    (module): AWD_LSTM(
      (encoder): Embedding(32256, 400, padding_idx=1)
      (encoder_dp): EmbeddingDropout(
        (emb): Embedding(32256, 400, padding_idx=1)
      )
```

```
        (rnns): ModuleList(
          (0): WeightDropout(
            (module): LSTM(400, 1150, batch_first=True)
          )
          (1): WeightDropout(
            (module): LSTM(1150, 1150, batch_first=True)
          )
          (2): WeightDropout(
            (module): LSTM(1150, 400, batch_first=True)
          )
        )
        (input_dp): RNNDropout()
        (hidden_dps): ModuleList(
          (0): RNNDropout()
          (1): RNNDropout()
          (2): RNNDropout()
        )
      )
    )
    (1): PoolingLinearClassifier(
      (layers): Sequential(
        (0): BatchNorm1d(1200, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
        (1): Dropout(p=0.2, inplace=False)
        (2): Linear(in_features=1200, out_features=50, bias=True)
        (3): ReLU(inplace=True)
        (4): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_
running_stats=True)
        (5): Dropout(p=0.1, inplace=False)
        (6): Linear(in_features=50, out_features=2, bias=True)
      )
    )
), opt_func=functools.partial(<class 'torch.optim.adam.Adam'>, betas=
(0.9, 0.99)), loss_func=FlattenedLoss of CrossEntropyLoss(), metrics=[<
function accuracy at 0x7f471f0168c8>], true_wd=True, bn_wd=True, wd=0.0
1, train_bn=True, path=PosixPath('.'), model_dir='models', callback_fns
=[functools.partial(<class 'fastai.basic_train.Recorder'>, add_time=Tru
e, silent=False)], callbacks=[...], layer_groups=[Sequential(
  (0): Embedding(32256, 400, padding_idx=1)
  (1): EmbeddingDropout(
    (emb): Embedding(32256, 400, padding_idx=1)
  )
), Sequential(
  (0): WeightDropout(
    (module): LSTM(400, 1150, batch_first=True)
  )
  (1): RNNDropout()
), Sequential(
  (0): WeightDropout(
    (module): LSTM(1150, 1150, batch_first=True)
  )
  (1): RNNDropout()
), Sequential(
  (0): WeightDropout(
    (module): LSTM(1150, 400, batch_first=True)
  )
  (1): RNNDropout()
), Sequential(
  (0): PoolingLinearClassifier(
    (layers): Sequential(
      (0): BatchNorm1d(1200, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
      (1): Dropout(p=0.2, inplace=False)
      (2): Linear(in_features=1200, out_features=50, bias=True)
      (3): ReLU(inplace=True)
      (4): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_
running_stats=True)
      (5): Dropout(p=0.1, inplace=False)
      (6): Linear(in_features=50, out_features=2, bias=True)
    )
  )
)], add_time=True, silent=False)
alpha: 2.0
beta: 1.0, MixedPrecision
learn: RNNLearner(data=TextClasDataBunch;

Train: LabelList (158404 items)
x: TextList
```

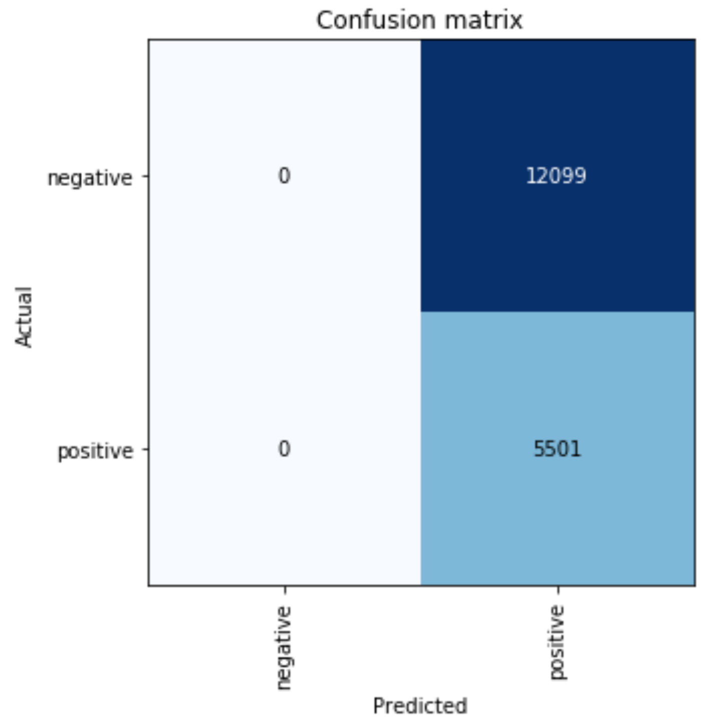
```
xxbos xxmaj se imaginan a los chicos agradeciendo por el premio con car
a de xxunk bonito . xxmaj one xxmaj direction,xxbos xxmaj xxunk - 12 xx
maj siempre , promesa,xxbos xxmaj qué xxunk , xxup pj . ya no compartes
ni un xxunk con nosotros .,xxbos xxmaj buenos dias para todos . xxmaj f
eliz inicio de semana .,xxbos xxmaj gracias ! xxmaj no es así , deja cl
aro que es el 100 % de aquí }
y: CategoryList
positive,positive,positive,positive,positive
Path: .;

Valid: LabelList (17600 items)
x: TextList
xxbos xxmaj lo extraño tanto ( (,xxbos xxmaj buenos días hoy toca ^_^,x
xbos xxmaj mi estado de animo ahora mismo es una mierda . xxmaj por sue
rte tengo a unas amigas que me han reír con sus estupideces y eso me ay
uda .,xxbos xxmaj necesito urgentemente que alguien me xxunk , no se qu
é hacer ..,xxbos xxmaj no me abandonen
y: CategoryList
negative,positive,positive,negative,negative
Path: .;

Test: None, model=SequentialRNN(
  (0): MultiBatchEncoder(
    (module): AWD_LSTM(
      (encoder): Embedding(32256, 400, padding_idx=1)
      (encoder_dp): EmbeddingDropout(
        (emb): Embedding(32256, 400, padding_idx=1)
      )
      (rnns): ModuleList(
        (0): WeightDropout(
          (module): LSTM(400, 1150, batch_first=True)
        )
        (1): WeightDropout(
          (module): LSTM(1150, 1150, batch_first=True)
        )
        (2): WeightDropout(
          (module): LSTM(1150, 400, batch_first=True)
        )
      )
```

```
        (input_dp): RNNDropout()
        (hidden_dps): ModuleList(
          (0): RNNDropout()
          (1): RNNDropout()
          (2): RNNDropout()
        )
      )
    )
    (1): PoolingLinearClassifier(
      (layers): Sequential(
        (0): BatchNorm1d(1200, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
        (1): Dropout(p=0.2, inplace=False)
        (2): Linear(in_features=1200, out_features=50, bias=True)
        (3): ReLU(inplace=True)
        (4): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_
running_stats=True)
        (5): Dropout(p=0.1, inplace=False)
        (6): Linear(in_features=50, out_features=2, bias=True)
      )
    )
), opt_func=functools.partial(<class 'torch.optim.adam.Adam'>, betas=
(0.9, 0.99)), loss_func=FlattenedLoss of CrossEntropyLoss(), metrics=[<
function accuracy at 0x7f471f0168c8>], true_wd=True, bn_wd=True, wd=0.0
1, train_bn=True, path=PosixPath('.'), model_dir='models', callback_fns
=[functools.partial(<class 'fastai.basic_train.Recorder'>, add_time=Tru
e, silent=False)], callbacks=[...], layer_groups=[Sequential(
  (0): Embedding(32256, 400, padding_idx=1)
  (1): EmbeddingDropout(
    (emb): Embedding(32256, 400, padding_idx=1)
  )
), Sequential(
  (0): WeightDropout(
    (module): LSTM(400, 1150, batch_first=True)
  )
  (1): RNNDropout()
), Sequential(
  (0): WeightDropout(
    (module): LSTM(1150, 1150, batch_first=True)
    )
    (1): RNNDropout()
  ), Sequential(
    (0): WeightDropout(
      (module): LSTM(1150, 400, batch_first=True)
    )
    (1): RNNDropout()
  ), Sequential(
    (0): PoolingLinearClassifier(
      (layers): Sequential(
        (0): BatchNorm1d(1200, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
        (1): Dropout(p=0.2, inplace=False)
        (2): Linear(in_features=1200, out_features=50, bias=True)
        (3): ReLU(inplace=True)
        (4): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_
running_stats=True)
        (5): Dropout(p=0.1, inplace=False)
        (6): Linear(in_features=50, out_features=2, bias=True)
      )
    )
  )], add_time=True, silent=False)
loss_scale: 65536
max_noskip: 1000
dynamic: True
clip: None
flat_master: False
max_scale: 16777216
loss_fp32: True], layer_groups=[Sequential(
    (0): Embedding(32256, 400, padding_idx=1)
    (1): EmbeddingDropout(
      (emb): Embedding(32256, 400, padding_idx=1)
    )
  ), Sequential(
    (0): WeightDropout(
      (module): LSTM(400, 1150, batch_first=True)
    )
    (1): RNNDropout()
  ), Sequential(
    (0): WeightDropout(
      (module): LSTM(1150, 1150, batch_first=True)
    )
    (1): RNNDropout()
  ), Sequential(
    (0): WeightDropout(
      (module): LSTM(1150, 400, batch_first=True)
    )
    (1): RNNDropout()
  ), Sequential(
    (0): PoolingLinearClassifier(
      (layers): Sequential(
        (0): BatchNorm1d(1200, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
        (1): Dropout(p=0.2, inplace=False)
        (2): Linear(in_features=1200, out_features=50, bias=True)
        (3): ReLU(inplace=True)
        (4): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_
running_stats=True)
        (5): Dropout(p=0.1, inplace=False)
        (6): Linear(in_features=50, out_features=2, bias=True)
      )
    )
)], add_time=True, silent=False)
```

We can see that an untrained model does a terrible job of classifying tweets:

In [108]:
```
# Print out the current state of the model
interp = ClassificationInterpretation.from_learner(learn_clas)
print('Confusion Matrix:')
print(interp.confusion_matrix())
interp.plot_confusion_matrix(figsize=(5,5))
# interp.plot_top_losses(9, figsize=(15,11))
```

```
Confusion Matrix:
[[    0 12099]
 [    0  5501]]
```



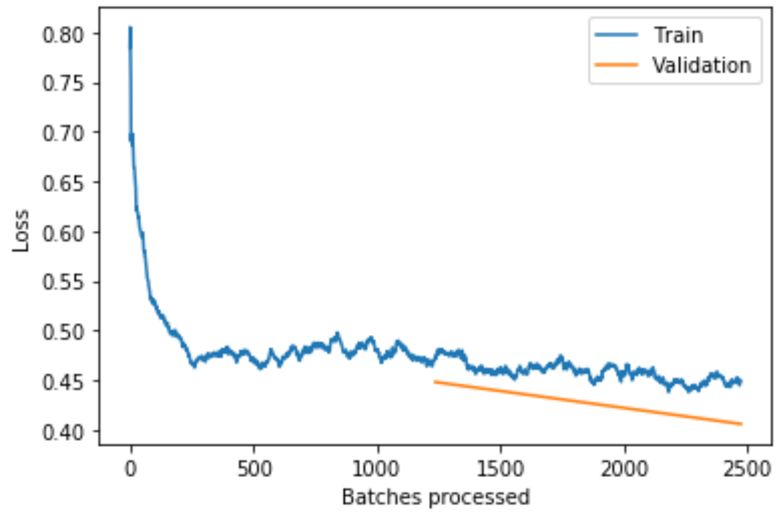
```
In [0]: # freeze the internal layers
        learn_clas.freeze()
```

```
In [0]: lr=2e-2
        lr *= bs/48
```

```
In [111]: learn_clas.fit_one_cycle(2, lr, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.471394	0.447979	0.794830	00:44
1	0.449237	0.405826	0.818125	00:45

```
In [112]: learn_clas.recorder.plot_losses()
```

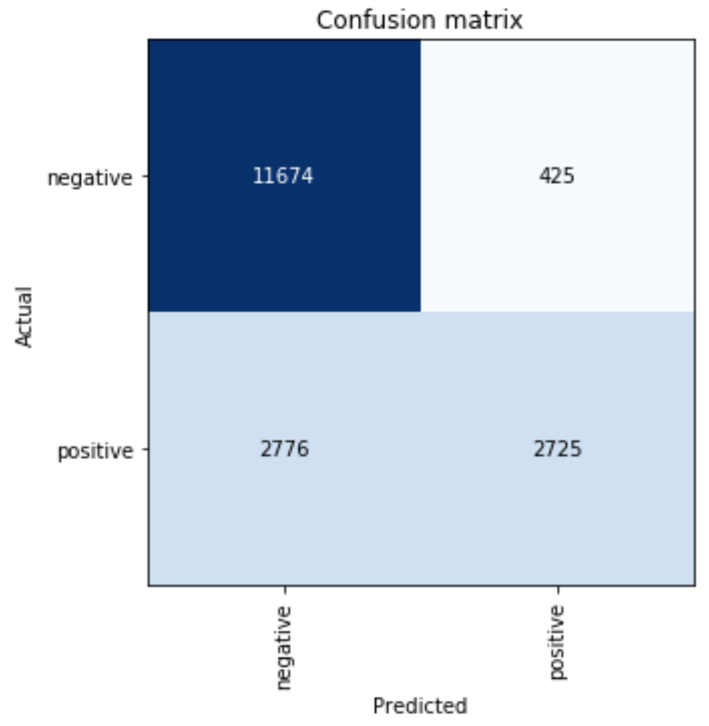


In [0]:
```
learn_clas.save(model_path/'tweet_classifier_1')
```

In [0]:
```
learn_clas.load(model_path/'tweet_classifier_1');
```

In [115]:
```
# Print out the current state of the model
interp = ClassificationInterpretation.from_learner(learn_clas)
print('Confusion Matrix:')
print(interp.confusion_matrix())
interp.plot_confusion_matrix(figsize=(5,5))
# interp.plot_top_losses(9, figsize=(15,11))
```

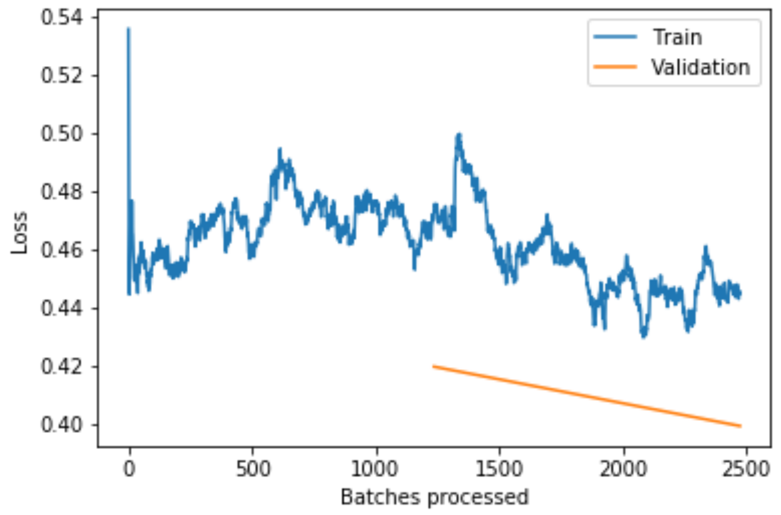
```
Confusion Matrix:
[[11674   425]
 [ 2776  2725]]
```



```
In [116]: learn_clas.fit_one_cycle(2, lr, moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.470782	0.419609	0.812443	00:45
1	0.445360	0.399272	0.822159	00:45

```
In [117]: learn_clas.recorder.plot_losses()
```



```
In [0]: learn_clas.save(model_path/'tweet_classifier_2')
```

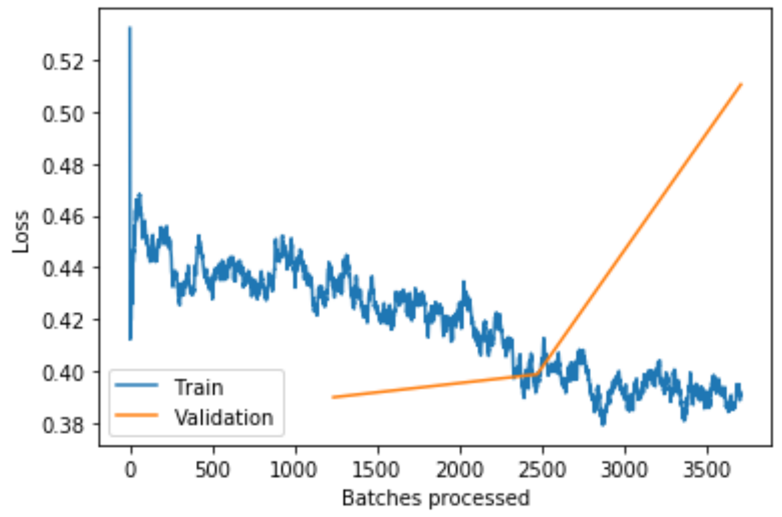
```
In [0]: learn_clas.load(model_path/'tweet_classifier_2');
```

Now that further training isn't improving the validation error, we can start un-freezing more layers and training them. Note how we reduce the learning rate every time we unfreeze more layers. The specific parameters used here were determined empirically by the people at `fastai`, and they work well for language training:

```
In [120]: learn_clas.freeze_to(-2)
          learn_clas.fit_one_cycle(3, slice(lr/(2.6**4),lr), moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.436105	0.389774	0.829034	00:57
1	0.393454	0.398626	0.833125	00:56
2	0.391793	0.510550	0.841648	00:52

```
In [121]: learn_clas.recorder.plot_losses()
```



Whoa, what happened here? Load up a previous state!

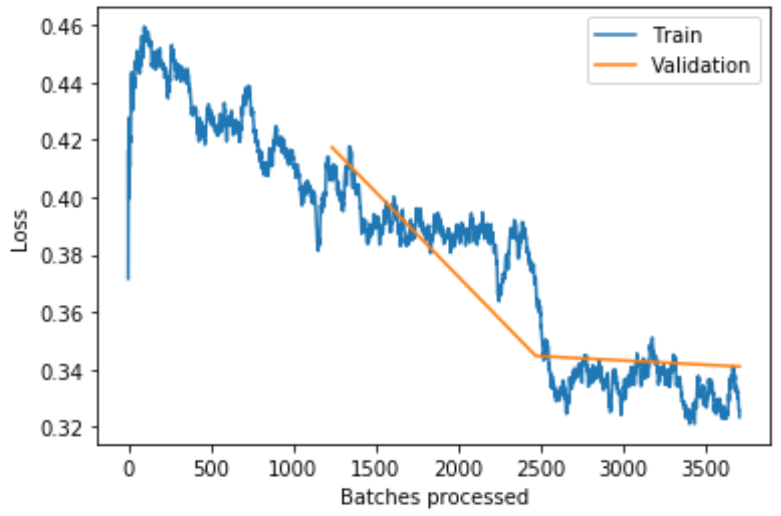
In [0]:
```
# learn_clas.save(model_path/'tweet_classifier_3')
```

In [0]:
```
# Recover the previous model
learn_clas.load(model_path/'tweet_classifier_2');
```

In [126]:
```
learn_clas.freeze_to(-3)
learn_clas.fit_one_cycle(3, slice(lr/2/(2.6**4),lr/2), moms=(0.8,0.7))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.405964	0.417393	0.829489	01:17
1	0.370628	0.344733	0.849261	01:15
2	0.323364	0.341040	0.853750	01:17

In [127]:
```
learn_clas.recorder.plot_losses()
```



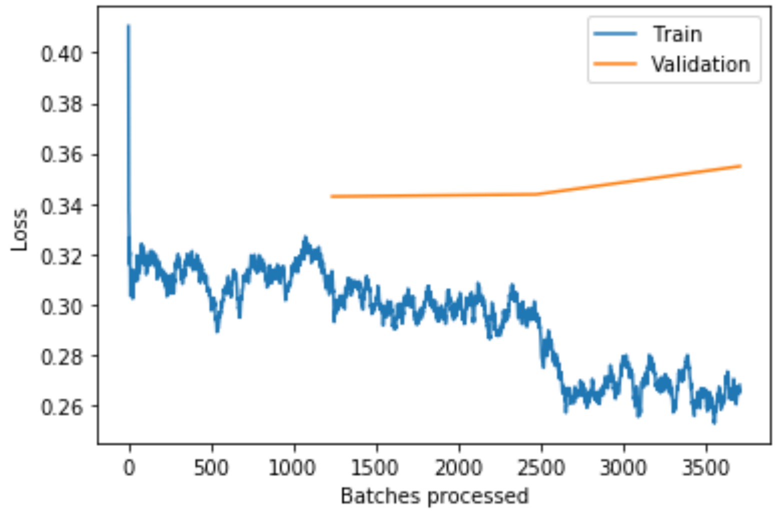
In [0]:
```
learn_clas.save(model_path/'tweet_classifier_4')
```

In [0]:
```
learn_clas.load(model_path/'tweet_classifier_4');
```

In [129]:
```
# Unfreeze the whole model
learn_clas.unfreeze()
learn_clas.fit_one_cycle(3, slice(lr/10/(2.6**4),lr/10), moms=(0.8,0.7
))
```

epoch	train_loss	valid_loss	accuracy	time
0	0.312630	0.342901	0.851591	01:40
1	0.294511	0.343735	0.853977	01:38
2	0.268072	0.354890	0.852955	01:46

In [130]:
```
learn_clas.recorder.plot_losses()
```



In [0]:
```
learn_clas.save(model_path/'tweet_classifier_5');
```

In [0]:
```
learn_clas.load(model_path/'tweet_classifier_5');
```

Now that our model has stopped improving, we're done!

Interpreting our tweet classifier (20 mts)

We can use interpretation methods to see what parts of a tweet are drawing the model's attention; i.e. which words are more important for the model in making its decisions:

In [0]:
```
import matplotlib.cm as cm
import warnings
warnings.filterwarnings('ignore')
```

In [0]:
```
interp = TextClassificationInterpretation.from_learner(learn_clas)
```

In [136]:
```
test_text = "Domingo: .- ¡Que emoción, ya quiero entrar a clases, con t
```

```
odo este inicio!\
              ..Lunes:. - ¿Cuando acaba el semestre?"

print(learn_clas.predict(test_text))
print()
print(test_text)
interp.show_intrinsic_attention(test_text, cmap=cm.Reds);
```

```
(Category negative, tensor(0), tensor([0.9159, 0.0841]))

Domingo: .- ¡Que emoción, ya quiero entrar a clases, con todo este inic
io!              ..Lunes:. - ¿Cuando acaba el semestre?
```

```
xxbos xxmaj domingo : .- ¡ xxmaj que emoción , ya quiero entrar a
clases , con todo este inicio ! .. xxmaj lunes : . - ¿ xxmaj cuando
acaba el semestre ?
```

In [137]:
```
test_text = "El primer poema que leí fue la sonrisa de mi madre.."


print(learn_clas.predict(test_text))
print()
print(test_text)
interp.show_intrinsic_attention(test_text, cmap=cm.Greens);
```

```
(Category positive, tensor(1), tensor([0.4612, 0.5388]))

El primer poema que leí fue la sonrisa de mi madre..
```

```
xxbos xxmaj el primer poema que leí fue la sonrisa de mi madre ..
```

In [138]:
```
interp.intrinsic_attention(test_text)[1]
```

Out[138]:
```
tensor([0.0115, 0.0525, 0.1440, 0.4270, 0.6958, 0.2054, 0.4841, 0.4143,
0.3228,
        1.0000, 0.2446, 0.1816, 0.2759, 0.1613], device='cuda:0',
       dtype=torch.float16)
```

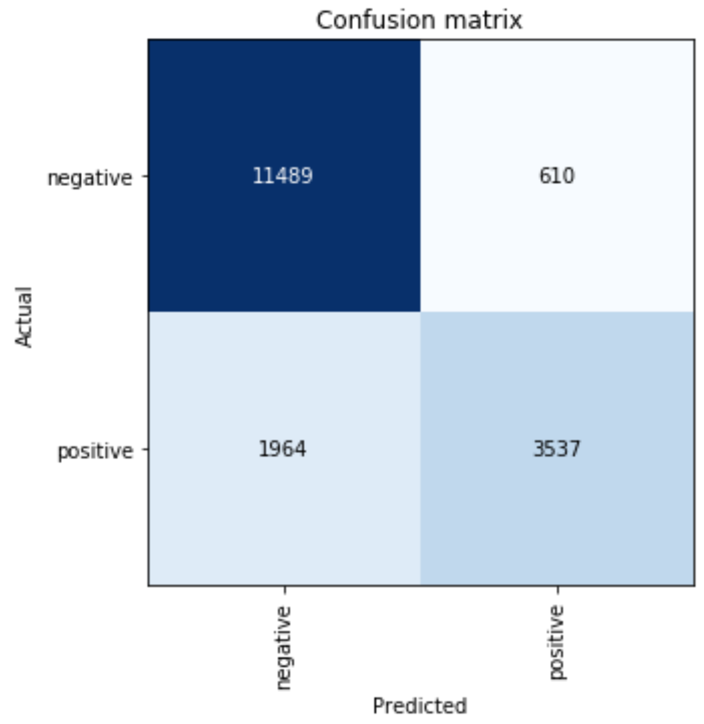
In [139]:
```
interp.show_top_losses(20)
```

Text	Prediction	Actual	Loss	Probability
xxbos xxmaj esta noche no me toco xxunk como está jugando xxup clg . xxmaj brillante . xxmaj hasta xxmaj xxunk está xxunk en este game .. xxmaj ay	negative	positive	13.14	0.00
xxbos xxup jimmy xxup xfavor xxup di xxup algo . xxup xxunk ; (((	negative	positive	8.93	0.00
xxbos xxmaj mi abuelo me regalo dos remeras , una de xxmaj xxunk y la otra de xxmaj xxunk . >	negative	positive	7.81	0.00
xxbos xxmaj odio que los xxunk de menos de 5 o 4 años xxunk a gritar o algo asi , me molesta xxup demasiado	negative	positive	7.41	0.00
xxbos xxmaj yo se que algún me estaré riendo de estos malos momentos que hoy me hacen mal	negative	positive	6.87	0.00
xxbos xxmaj holi (	negative	positive	6.46	0.00
xxbos xxmaj cuando mire por la ventana al creer oir tu voz y no te vi me di cuenta de lo mucho que te extraño . j > / xxunk (	negative	positive	6.44	0.00
xxbos xxmaj todo mal	negative	positive	6.28	0.00
xxbos xxmaj por qué todo pasa en las tardes ?	negative	positive	6.21	0.00
xxbos xxmaj no se que hacer	negative	positive	6.18	0.00
xxbos tienes la entrevista porfi xxunk que no la pude ver	negative	positive	6.00	0.00
xxbos xxmaj extrañoo a xxmaj lauti , seguire siendo su xxunk ? xxmaj ah .te extraño xxmaj lautaro xxmaj xxunk , mii xxunk	negative	positive	5.79	0.00
xxbos xxmaj los recuerdos tristes viven en nuestro corazón , xxunk a valorar esos pequeños instantes de alegría	positive	negative	5.20	0.01
xxbos xxmaj mi perra me ha roto el cargador de la xxup 3ds xxmaj necesito uno	negative	positive	4.94	0.01
xxbos xxmaj jajajajajajaja , y todavia a xxmaj maduro le qda xxunk ? ? ? xxmaj ese ni pelo debe tener .. xxmaj no será que xxunk la mandaron pa xxmaj colombia ?	negative	positive	4.87	0.01
xxbos xxmaj sueño regresa ya ! ! ! xxmaj te necesito	negative	positive	4.82	0.01
xxbos a pesar de todo xxmaj hoy fue un dia excelente	positive	negative	4.79	0.01
xxbos ¿ xxmaj qué le pasa al oro ?	negative	positive	4.79	0.01

Text	Prediction	Actual	Loss	Probability
xxbos xxmaj por que siento que cuando yo este re enganchado se va a re enganchar con migo , igual yo no soy segunda de nadie como me dijiste	negative	positive	4.63	0.01
xxbos ¡ xxmaj hoy juega el mejor equipo del xxunk , el mejor equipo de xxmaj europa .. xxmaj ok , el mejor equipo de xxmaj italia .. xxunk , ok ! xxmaj hoy juega el	positive	negative	4.61	0.01

In [140]:
```
# Print out the current state of the model
interp = ClassificationInterpretation.from_learner(learn_clas)
print('Confusion Matrix:')
print(interp.confusion_matrix())
interp.plot_confusion_matrix(figsize=(5,5))
# interp.plot_top_losses(9, figsize=(15,11))
```

```
Confusion Matrix:
[[11489   610]
 [ 1964  3537]]
```



Conclusions (5 mts)

In this case, we took an existing RNN already trained on a large corpus of Spanish text and retrained it to specifically deal with the nuances of Spanish tweets. We then built a classifier on our tweets to classify them by positive or negative sentiment, and interpreted the results.

Takeaways (5 mts)

It used to be that we would have to spend tens of thousands of hours training a language model. Now, we are able to easily re-train pre-existing RNNs to adapt to any specific use case we may have.