In [1]:

```
from collections import defaultdict
import numpy as np
import pandas as pd

import matplotlib_venn as venn
from matplotlib_venn import venn2, venn2_circles, venn3, venn3_circles
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
import nltk
from nltk.corpus import stopwords, wordnet
from nltk import pos_tag
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import regexp_tokenize, word_tokenize, RegexpTokenizer

nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

%load_ext autoreload
%autoreload 2
```

executed in 1.59s, finished 18:50:04 2021-06-07

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\elena\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\elena\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\elena\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [2]:

```
# These functions are taken from one of the NPL lectures at Flatiron School

sw = stopwords.words('english')
def get_wordnet_pos(treebank_tag):
    '''
    Translate nltk POS to wordnet tags
    '''
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

def doc_preparer(doc, stop_words=sw):
    '''

    :param doc: a document from the satire corpus
    :return: a document string with words which have been
            lemmatized,
            parsed for stopwords,
            made lowercase,
            and stripped of punctuation and numbers.
    '''

    regex_token = RegexpTokenizer(r"([a-zA-Z]+(?:’[a-z]+)?)")
    doc = regex_token.tokenize(doc)
    doc = [word.lower() for word in doc]
    doc = [word for word in doc if word not in sw]
    print(doc)
    doc = pos_tag(doc)
    doc = [(word[0], get_wordnet_pos(word[1])) for word in doc]
    lemmatizer = WordNetLemmatizer()
    doc = [lemmatizer.lemmatize(word[0], word[1]) for word in doc]
    return ' '.join(doc)
```

executed in 62ms, finished 18:50:04 2021-06-07

In [3]:

```
# This part of the code is specific to the venn diagram demo of corpus word split
corpus = pd.read_csv('satire_nosatire.csv')

X = corpus.body
y = corpus.target

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    random_state=42,
                                                    test_size=0.25)

token_docs = [doc_preparer(doc, sw) for doc in X_train]

X_t, X_val, y_t, y_val = train_test_split(token_docs, y_train,
                                          test_size=0.25, random_state=42)

df_X_t=pd.DataFrame(X_t)
df_y_t=pd.DataFrame(y_t)
df_train=pd.concat([df_X_t, df_y_t], axis=1, join="inner")
df_train.rename(columns={0: 'phrase'})

list_satire_phrases=[]
df_train_satire=df_train.loc[df_train.target==1]
L = [''.join(df_train_satire[x].astype(str)) for x in df_train_satire]
df_one_phrase = pd.DataFrame([L], columns=df_train_satire.columns)
df_one_phrase=df_one_phrase.rename(columns={0: 'long_phrase'})
satire_words=list(df_one_phrase.long_phrase)
list_of_satire_words=satire_words[0].split()
set_of_satire_words=set(list_of_satire_words)

list_news_phrases=[]
df_train_news=df_train.loc[df_train.target==0]
L = [''.join(df_train_news[x].astype(str)) for x in df_train_news]
df_one_phrase_news = pd.DataFrame([L], columns=df_train_news.columns)
df_one_phrase_news=df_one_phrase_news.rename(columns={0: 'long_phrase'})
news_words=list(df_one_phrase_news.long_phrase)
list_of_news_words=news_words[0].split()
set_of_news_words=set(list_of_news_words)
```

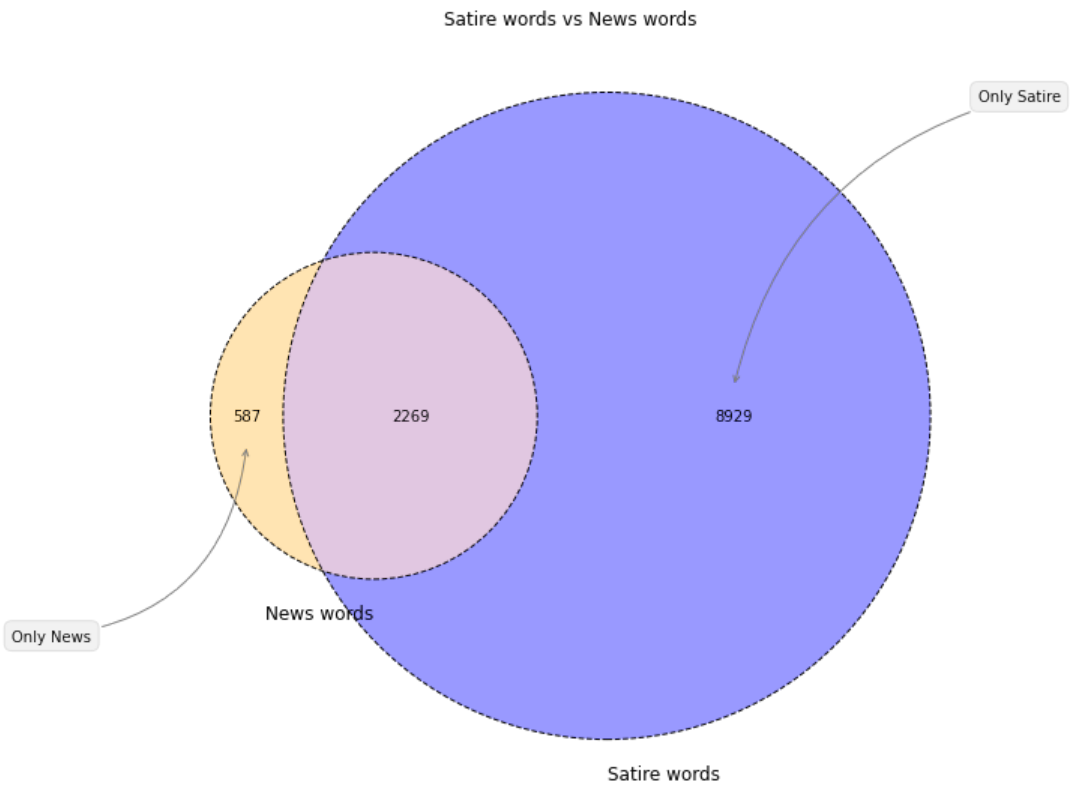
executed in 13.5s, finished 18:50:17 2021-06-07

```
['perpetually', 'offended', 'social', 'justice', 'warrior', 'keen', 'activist', 'socialist', 'matters', 'ranging', 'lgbtq
p', 'affairs', 'feminism', 'borders', 'soviet', 'ideology', 'marxism', 'ideal', 'communist', 'state', 'hugh', 'mungus',
'state', 'perpetual', 'offence', 'wake', 'morning', 'first', 'thing', 'hear', 'words', 'good', 'morning', 'shout', 'roomm
ate', 'form', 'racist', 'linguistic', 'capitalist', 'imperialist', 'sexist', 'white', 'male', 'created', 'offensive', 'gr
eeting', 'assumes', 'good', 'morning', 'western', 'capitalist', 'bourgeois', 'society', 'people', 'africa', 'good', 'morn
ing', 'shit', 'morning', 'live', 'corrugated', 'iron', 'shack', 'walk', 'five', 'hours', 'fill', 'bucket', 'full', 'dirt
y', 'muddy', 'water', 'fucking', 'drink', 'mr', 'mungus', 'attends', 'berkeley', 'college', 'california', 'outraged', 'de
ems', 'society', 'biology', 'science', 'offensive', 'well', 'concept', 'offended', 'offensive', 'outraged', 'offended',
'offence', 'offensive', 'hateful', 'perpetual', 'state', 'concept', 'offended', 'offending', 'offended', 'offensive', 'st
ance', 'state', 'offence', 'offending', 'offended', 'doubt', 'mind', 'creation', 'capitalist', 'racist', 'sexist', 'syste
m', 'imprisons', 'people', 'offended', 'suspect', 'created', 'offence', 'first', 'place', 'everything', 'offensive', 'per
petually', 'state', 'offence', 'right', 'offended', 'concept', 'offended', 'offence', 'offensive', 'manner']
['ana', 'luz', 'sister', 'law', 'ronald', 'blanco', 'looked', 'grimly', 'neighbours', 'murdered', 'honduran', 'man', 'was
hed', 'away', 'rills', 'blood', 'left', 'bullet', 'ridden', 'body', 'lain', 'outside', 'house', 'troubled', 'barrio', 'ou
tskirts', 'tegucigalpa', 'one', 'many', 'scenes', 'witnessed', 'year', 'assignment', 'honduras', 'thousands', 'people',
'sought', 'escape', 'violence', 'poverty', 'joining', 'migrant', 'caravan', 'hope', 'making', 'safety', 'across', 'mexic
o', 'u', 'border', 'problems', 'small', 'central', 'american', 'country', 'grabbed', 'international', 'attention', 'u',
'president', 'donald', 'trump', 'cracked', 'illegal', 'immigration', 'honduras', 'years', 'one', 'world's', 'murderous',
'countries', 'though', 'official', 'data', 'show', 'homicide', 'rate', 'fallen', 'sharply', 'continues', 'highly', 'chall
```

In [4]:

```
plt.figure(figsize=(10, 10))


sets=[set_of_news_words, set_of_satire_words]
labels=('News words', 'Satire words')

v=venn2(sets, set_labels = labels, set_colors=("orange", "blue"))

v.get_patch_by_id('10').set_alpha(0.3)


venn2_circles(subsets=sets,
              linestyle="dashed", linewidth=1)

plt.annotate('Only News',
             xy=v.get_label_by_id('10').get_position() - np.array([0, 0.05]), xytext=(-130,-130),
             ha='center', textcoords='offset points', bbox=dict(boxstyle='round, pad=0.5', fc='gray', alpha=0.1),
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=0.4',color='gray'))

plt.annotate('Only Satire',
             xy=v.get_label_by_id('01').get_position() - np.array([0, -0.05]), xytext=(190,190),
             ha='center', textcoords='offset points', bbox=dict(boxstyle='round, pad=0.5', fc='gray', alpha=0.1),
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=0.3',color='gray'))


plt.title('Satire words vs News words')
plt.show()
```

executed in 223ms, finished 18:50:18 2021-06-07



In [ ]:

```
1
```