In [1]:
```
#database final.sqlite is created after cleaning the amazon food reviews data

import sqlite3
import pandas as pd
con=sqlite3.connect("final.sqlite")
```

In [2]:
```
clean_reviews=pd.read_sql_query(""" Select * from Reviews """ , con)
clean_reviews=clean_reviews[:2000]
score=clean_reviews['Score']
```

In [3]:
```
from sklearn.feature_extraction.text import CountVectorizer
count_vect=CountVectorizer()
final_counts=count_vect.fit_transform(clean_reviews['CleanedText'].values)
final_counts.toarray()
type(final_counts)
final_counts.get_shape()
final_counts[0]
```

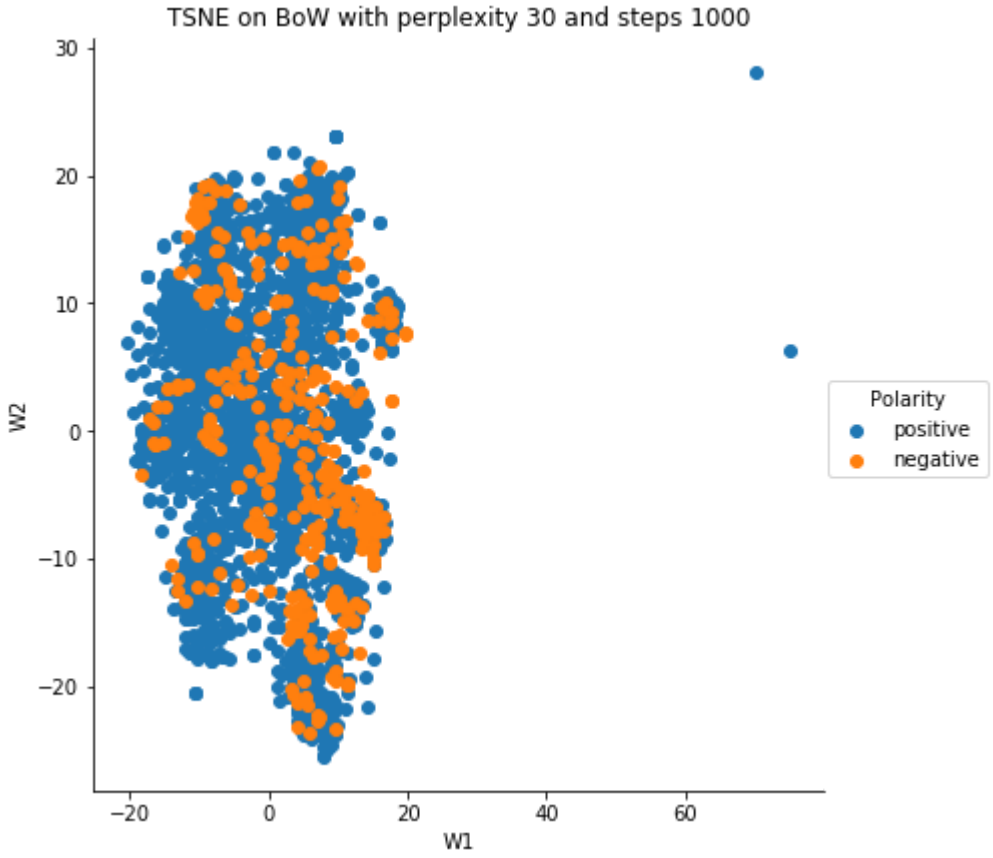
Out[3]:
```
<1x6858 sparse matrix of type '<class 'numpy.int64'>'
        with 31 stored elements in Compressed Sparse Row format>
```

In [4]:
```
from sklearn.manifold import TSNE
import scipy as sp
import seaborn as sns
import matplotlib.pyplot as plt


#TSNE with perplexity 30 and steps 1000

model=TSNE(n_components=2,random_state=0)
tsne_data=model.fit_transform(final_counts.toarray())
import numpy as np
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
```

In [5]:
```
#TSNE with perplexity 100 and steps 1000
```

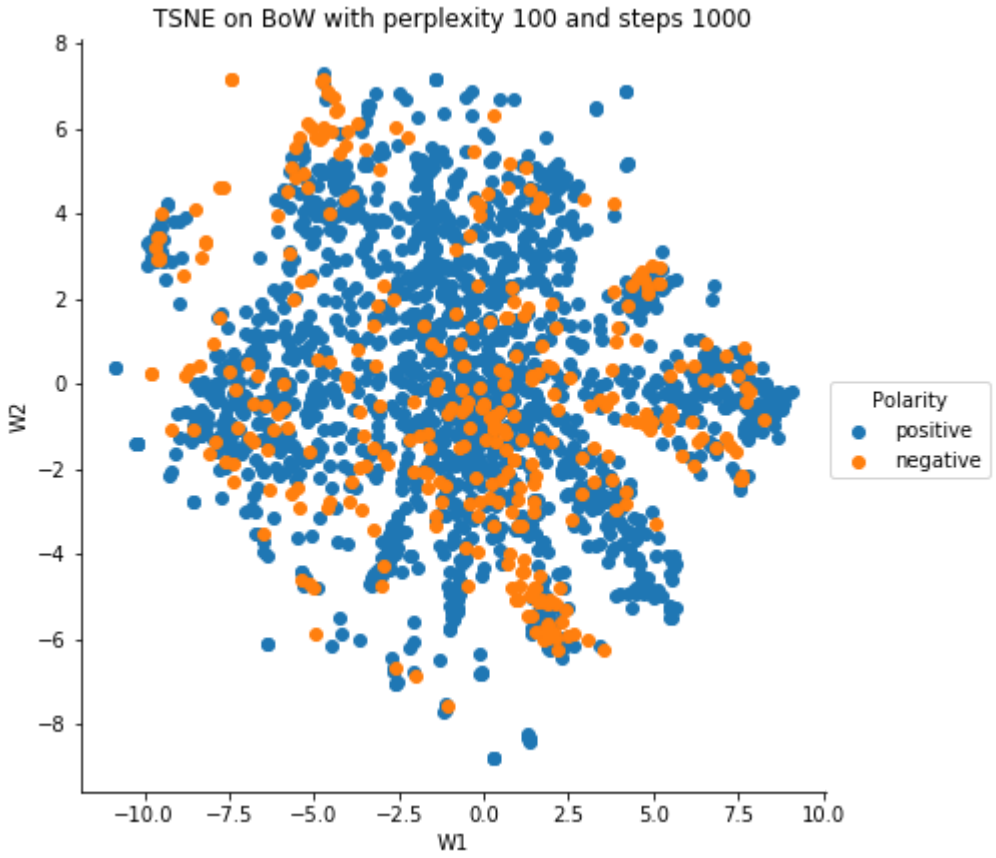
In [6]:

```
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on BoW with perplexity 30 and steps 1000")

plt.show()
```



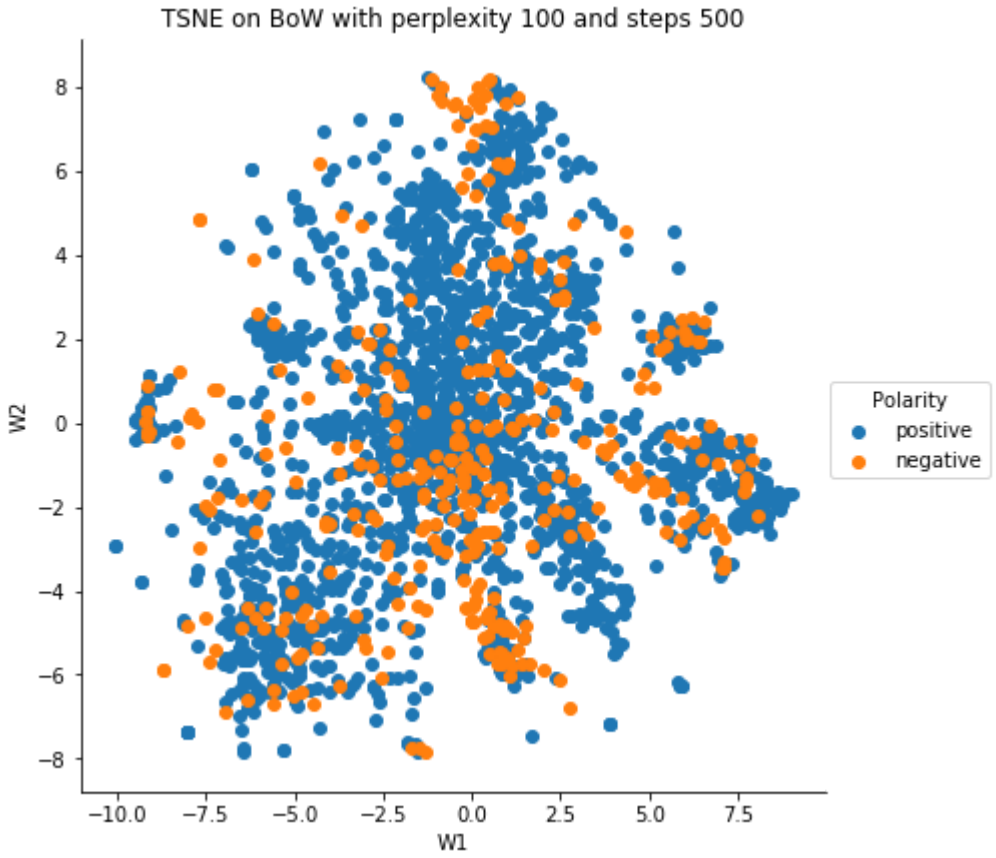
We could not get a clear separation of data : not linearly separable data::
lets try now with different perplexity and iterations

In [7]:
```
#TSNE with perplexity 100 and steps 1000
model=TSNE(n_components=2,random_state=0,perplexity=100)
tsne_data=model.fit_transform(final_counts.toarray())
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on BoW with perplexity 100 and steps 1000")
plt.show()
```

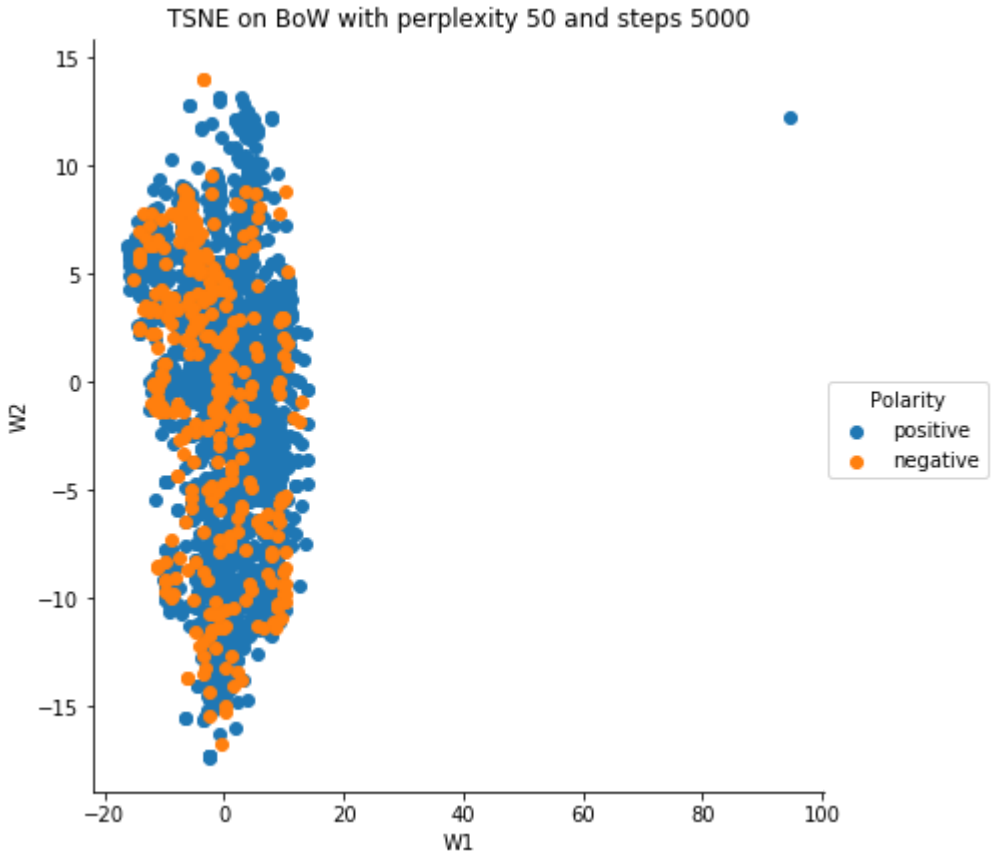


The plot gets more dispersed. No plane can be fit to separate the positive with negative reviews. Trying with changed perplexity and iterations

In [8]:
```
#TSNE with perplexity 100 and steps 500
model=TSNE(n_components=2,random_state=0,perplexity=100,n_iter=500)
tsne_data=model.fit_transform(final_counts.toarray())
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on BoW with perplexity 100 and steps 500")
plt.show()
```



In [9]:
```
#TSNE with perplexity 50 and steps 5000
model=TSNE(n_components=2,random_state=0,perplexity=50,n_iter=5000)
tsne_data=model.fit_transform(final_counts.toarray())
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on BoW with perplexity 50 and steps 5000")
plt.show()
```

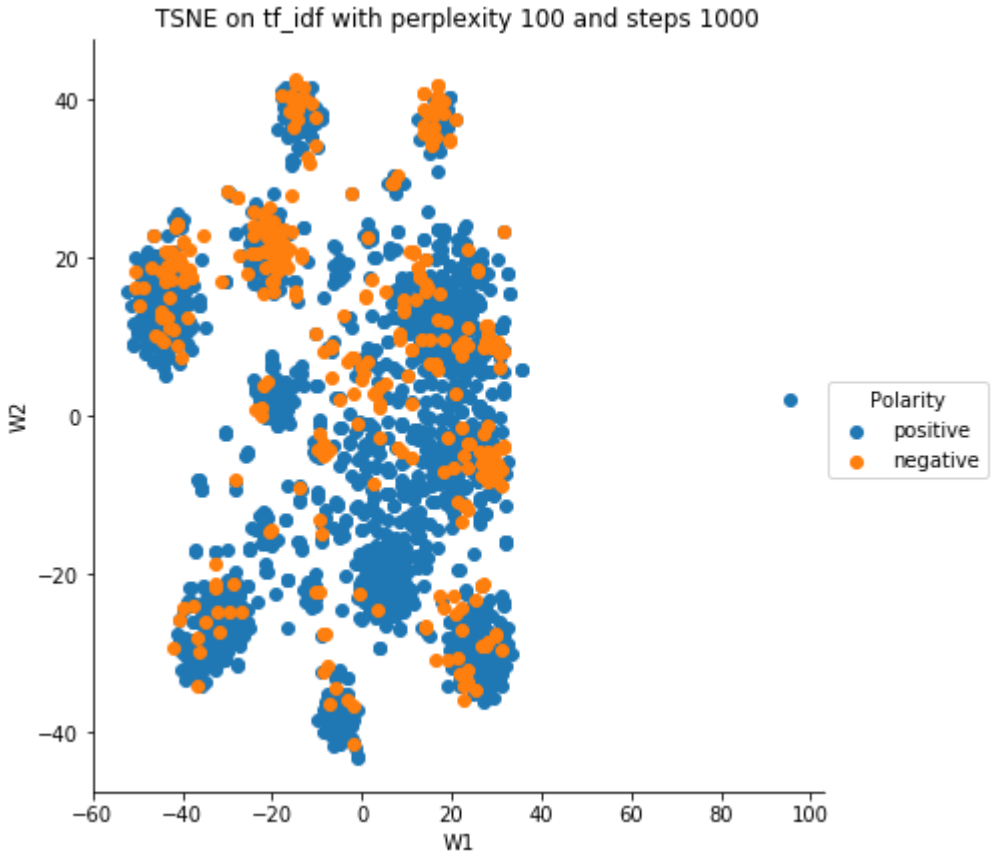


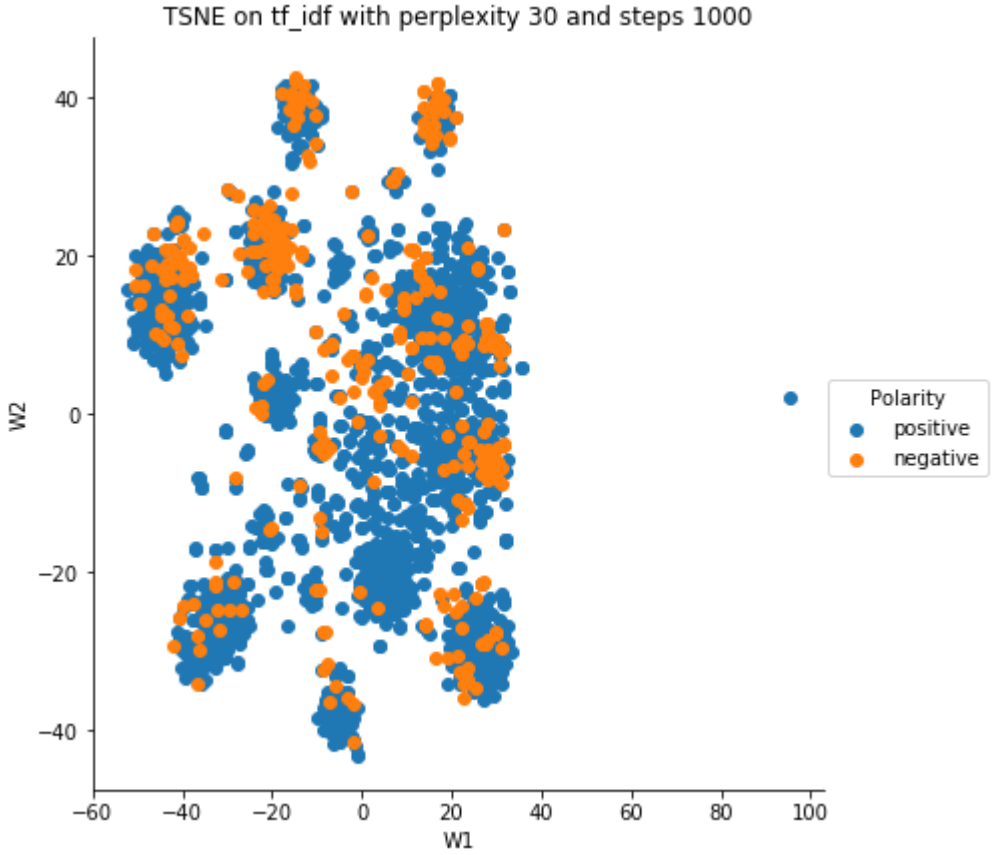
Changing perplexity and iterations for BoW also does not give us a clear picture of how to separate the positive and negative reviews geometrically. TSNE does not prove effective here.

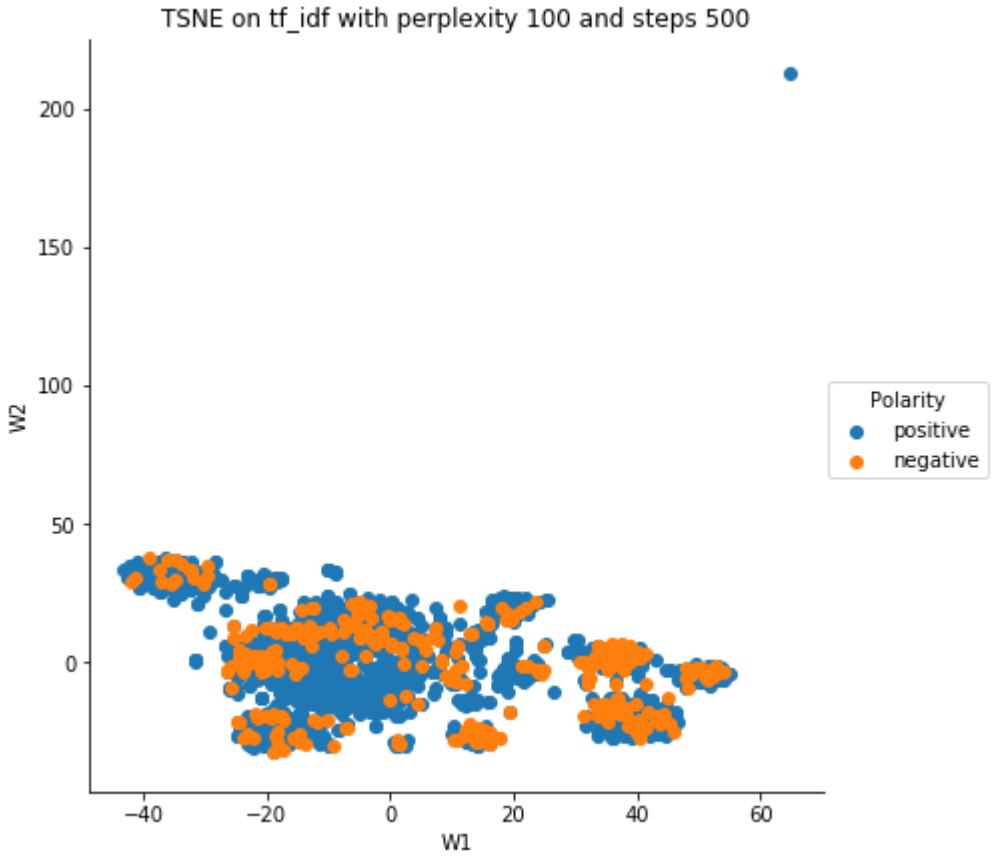
In [10]:
```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
tf_idf_vect = TfidfVectorizer()
final_counts= tf_idf_vect.fit_transform(clean_reviews['CleanedText'].values)
final_counts.toarray()
type(final_counts)
final_counts.get_shape()
final_counts[0]
```

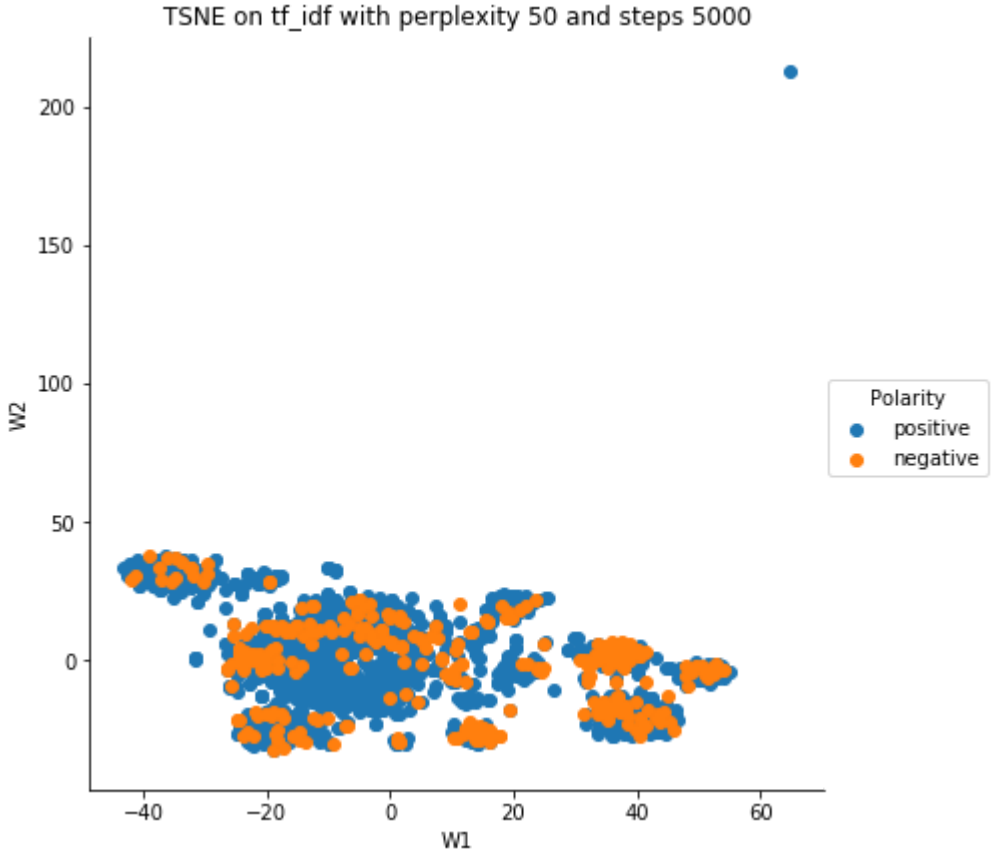
Out[10]:
```
<1x6858 sparse matrix of type '<class 'numpy.float64'>'
        with 31 stored elements in Compressed Sparse Row format>
```

In [11]:
```
#TSNE with perplexity 100 and steps 1000
model=TSNE(n_components=2,random_state=0)
tsne_data=model.fit_transform(final_counts.toarray())
import numpy as np
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on tf_idf with perplexity 100 and steps 1000")
plt.show()


#TSNE with perplexity 30 and steps 1000
model=TSNE(n_components=2,random_state=0,perplexity=30)
tsne_data=model.fit_transform(final_counts.toarray())
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on tf_idf with perplexity 30 and steps 1000")
plt.show()

#TSNE with perplexity 50 and steps 5000
model=TSNE(n_components=2,random_state=0,perplexity=50,n_iter=5000)
tsne_data=model.fit_transform(final_counts.toarray())
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on tf_idf with perplexity 100 and steps 500")
plt.show()

#TSNE with perplexity 50 and steps 5000
model=TSNE(n_components=2,random_state=0,perplexity=50,n_iter=5000)
tsne_data=model.fit_transform(final_counts.toarray())
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on tf_idf with perplexity 50 and steps 5000")
plt.show()
```


TSNE on tf_idf with perplexity 100 and steps 1000
W2
W1
Polarity
positive
negative


TSNE on tf_idf with perplexity 30 and steps 1000
W2
W1
Polarity
positive
negative





TSNE with different values of perplexity and iterations on tf-idf also does not help us demarcate geometrically a boundary wherein we can separate the polarity of the reviews

In [12]:
```
# Train your own Word2Vec model using your own text corpus
i=0
list_of_sent=[]
for sent in clean_reviews['CleanedText'].values:
    list_of_sent.append(sent.split())


print(clean_reviews['CleanedText'].values[0])
print(list_of_sent[0])
```

```
witti littl book make son laugh loud recit car drive along alway sing refrain h
es learn whale india droop love new word book introduc silli classic book will
bet son still abl recit memori colleg
['witti', 'littl', 'book', 'make', 'son', 'laugh', 'loud', 'recit', 'car', 'dri
ve', 'along', 'alway', 'sing', 'refrain', 'hes', 'learn', 'whale', 'india', 'dr
oop', 'love', 'new', 'word', 'book', 'introduc', 'silli', 'classic', 'book', 'w
ill', 'bet', 'son', 'still', 'abl', 'recit', 'memori', 'colleg']
```

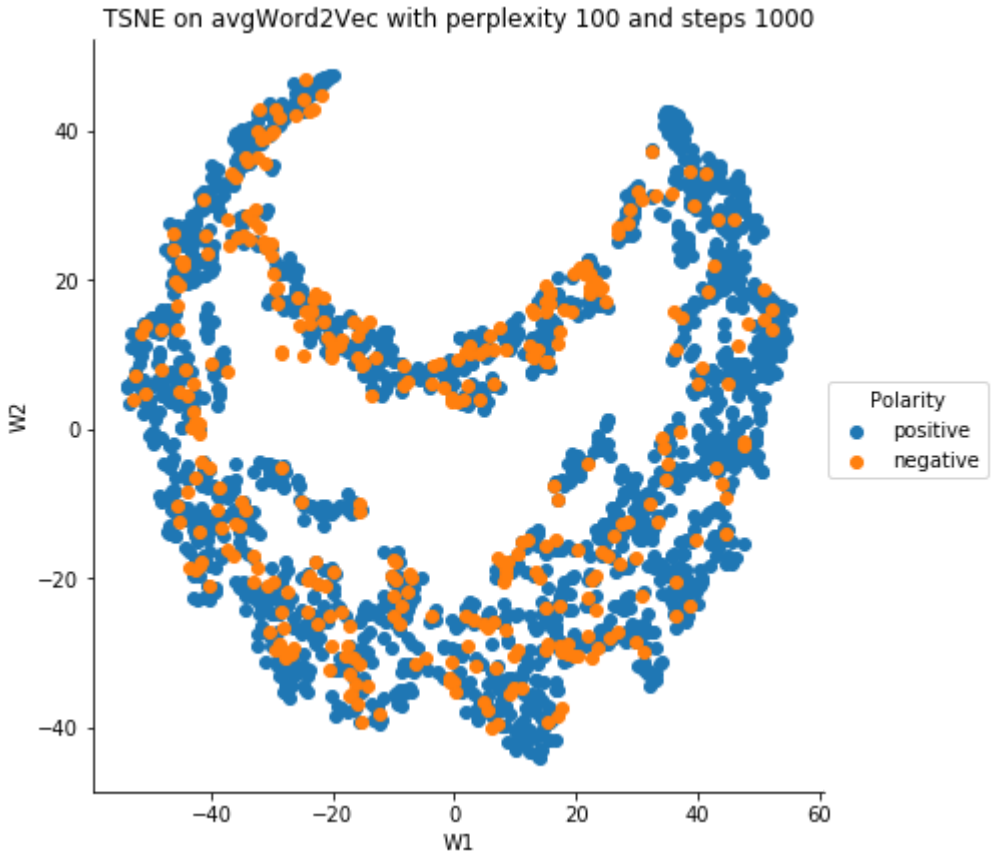
In [13]:
```
import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

w2v_model=Word2Vec(list_of_sent,min_count=5,size=50, workers=4)
```

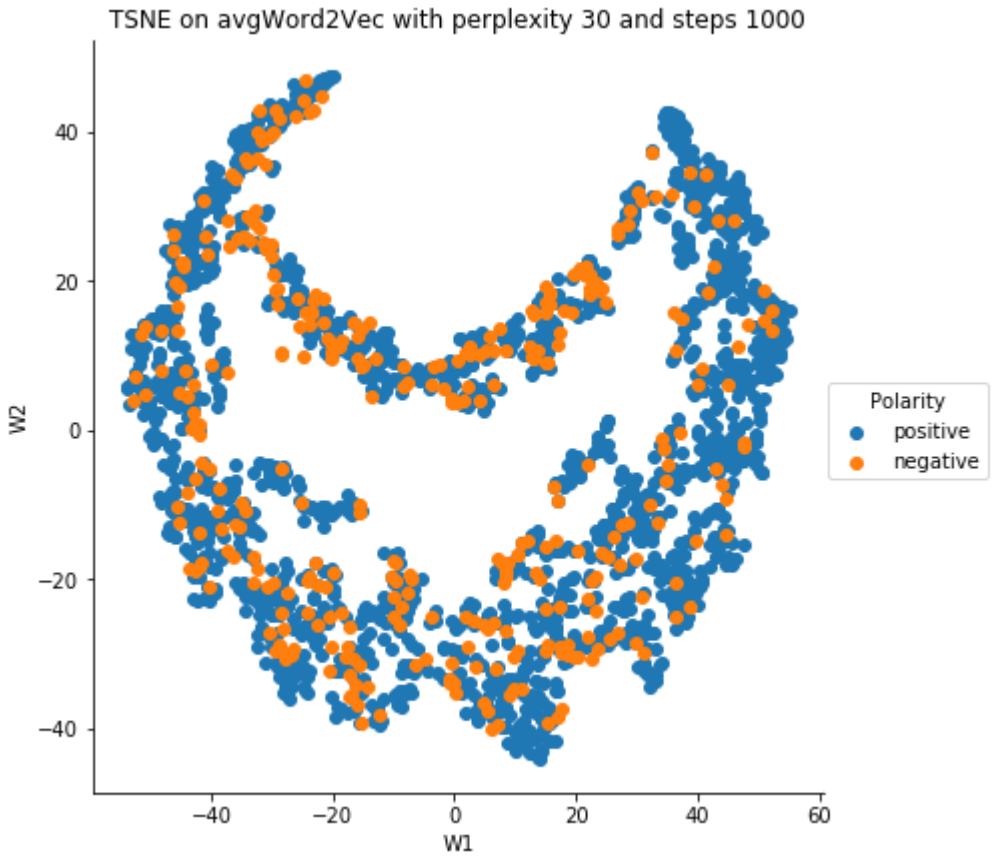
```
C:\ProgramData\Anaconda3\lib\site-packages\gensim\utils.py:1209: UserWarning: d
etected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

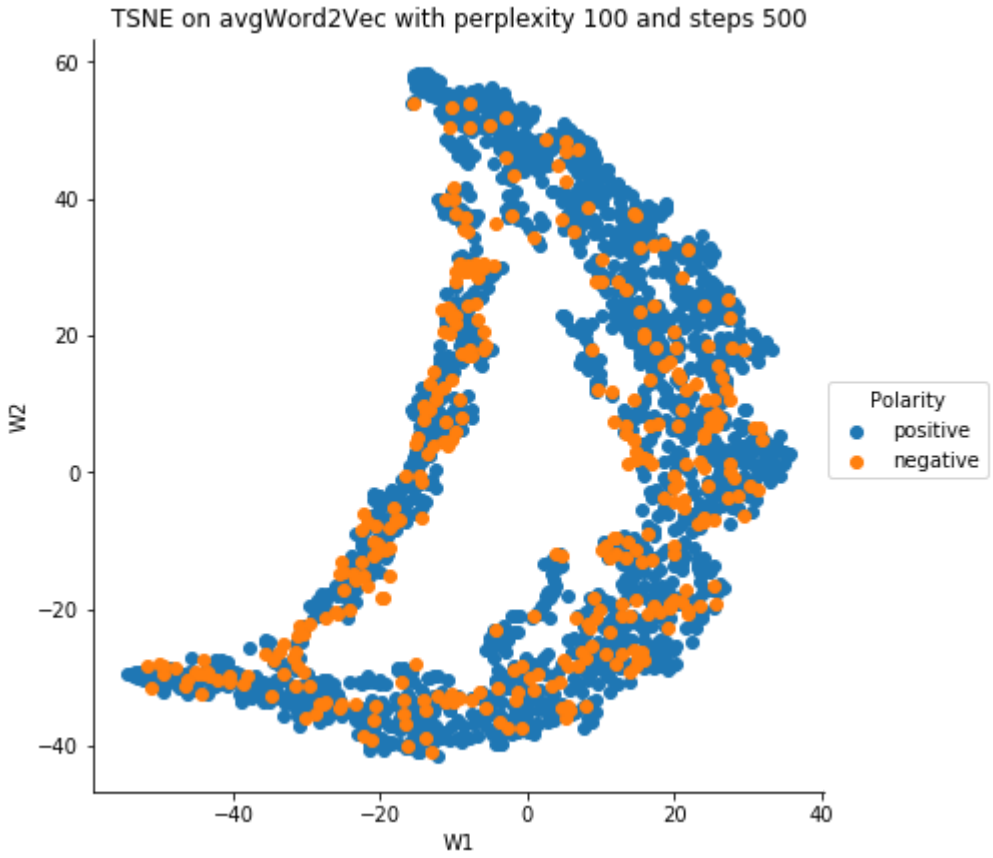
In [14]:
```
w2v_words = list(w2v_model.wv.vocab)
sent_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sent in list_of_sent: # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors.append(sent_vec)
print(len(sent_vectors))
print(len(sent_vectors[0]))
final_counts=sent_vectors
```

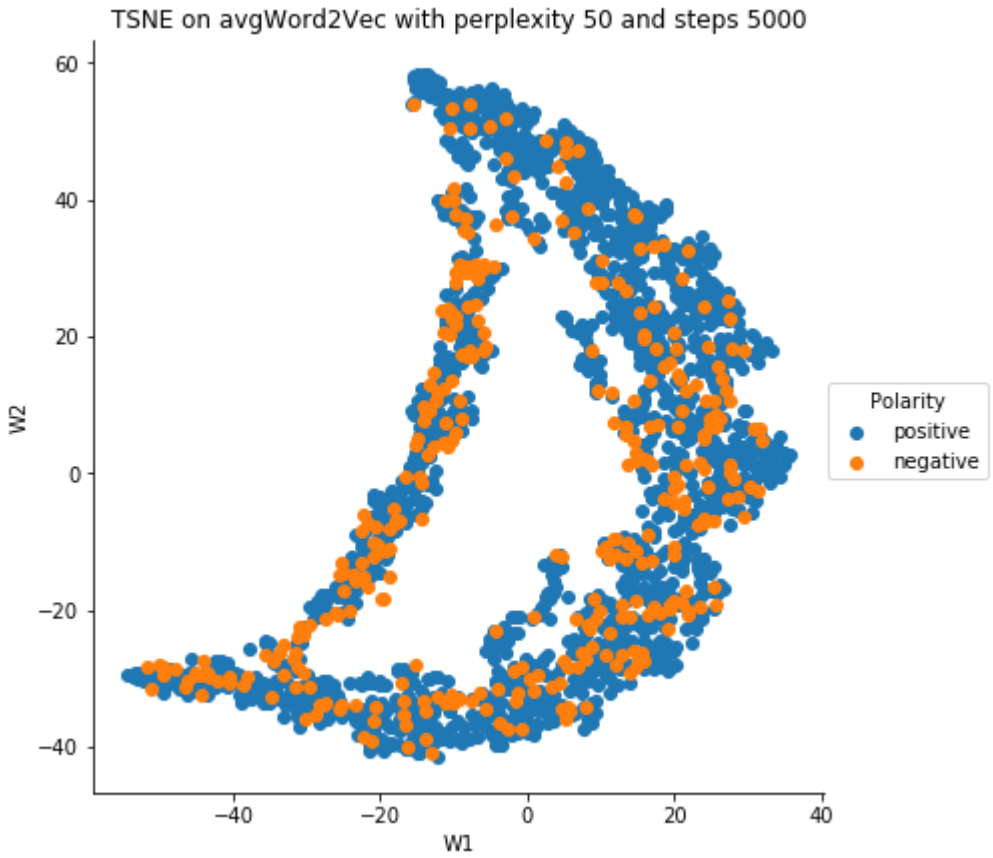
```
2000
50
```

In [15]:
```
#TSNE with perplexity 100 and steps 1000
model=TSNE(n_components=2,random_state=0)
tsne_data=model.fit_transform(final_counts)
import numpy as np
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on avgWord2Vec with perplexity 100 and steps 1000")
plt.show()


#TSNE with perplexity 30 and steps 1000
model=TSNE(n_components=2,random_state=0,perplexity=30)
tsne_data=model.fit_transform(final_counts)
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on avgWord2Vec with perplexity 30 and steps 1000")
plt.show()

#TSNE with perplexity 50 and steps 5000
model=TSNE(n_components=2,random_state=0,perplexity=50,n_iter=5000)
tsne_data=model.fit_transform(final_counts)
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on avgWord2Vec with perplexity 100 and steps 500")
plt.show()

#TSNE with perplexity 50 and steps 5000
model=TSNE(n_components=2,random_state=0,perplexity=50,n_iter=5000)
tsne_data=model.fit_transform(final_counts)
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on avgWord2Vec with perplexity 50 and steps 5000")
plt.show()
```


TSNE on avgWord2Vec with perplexity 100 and steps 1000
W2
W1
Polarity
positive
negative


TSNE on avgWord2Vec with perplexity 30 and steps 1000
W2
W1
Polarity
positive
negative

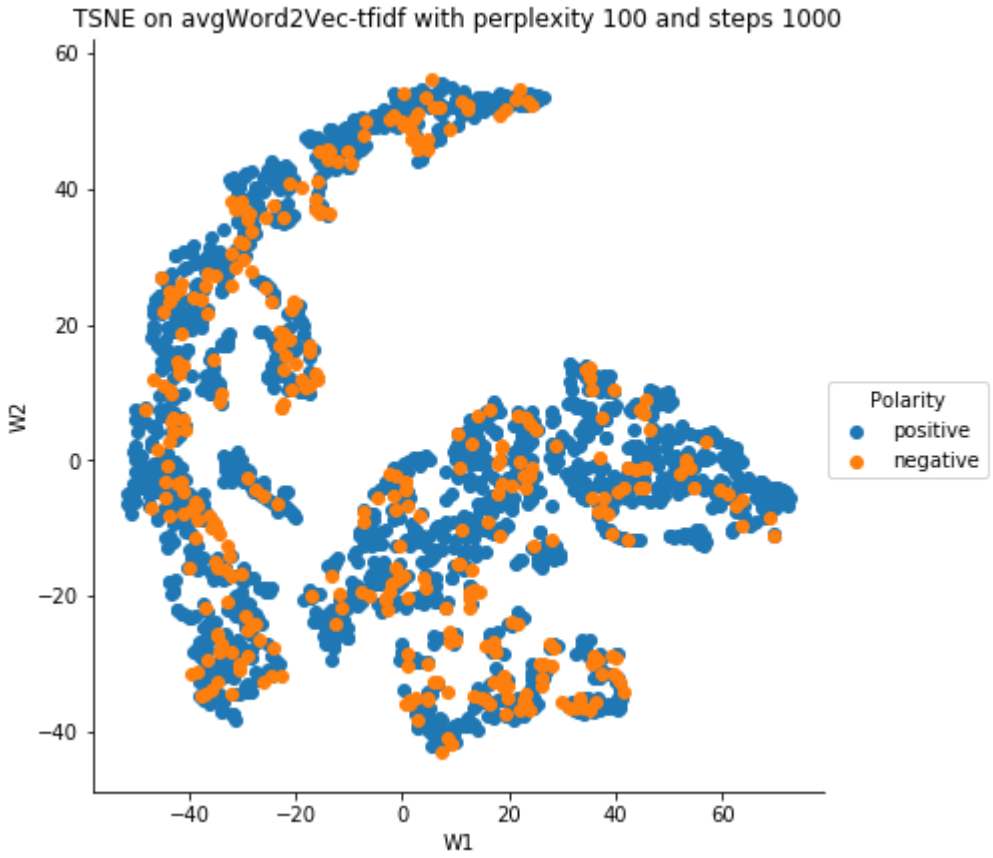


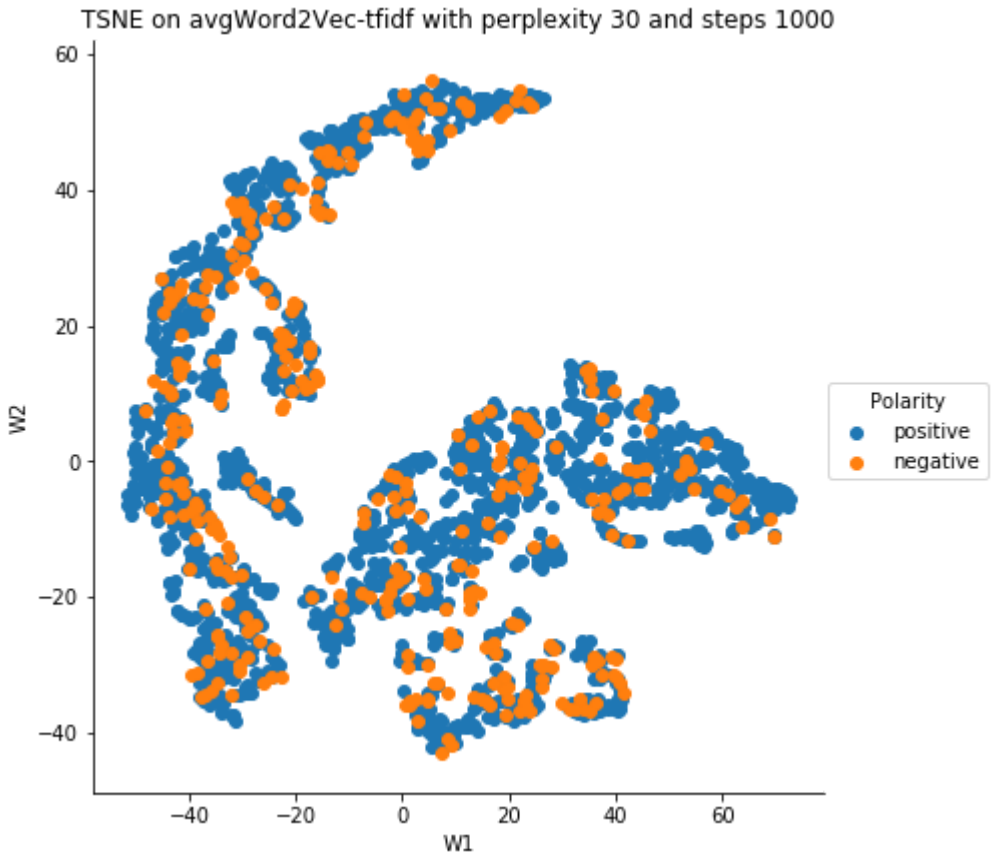


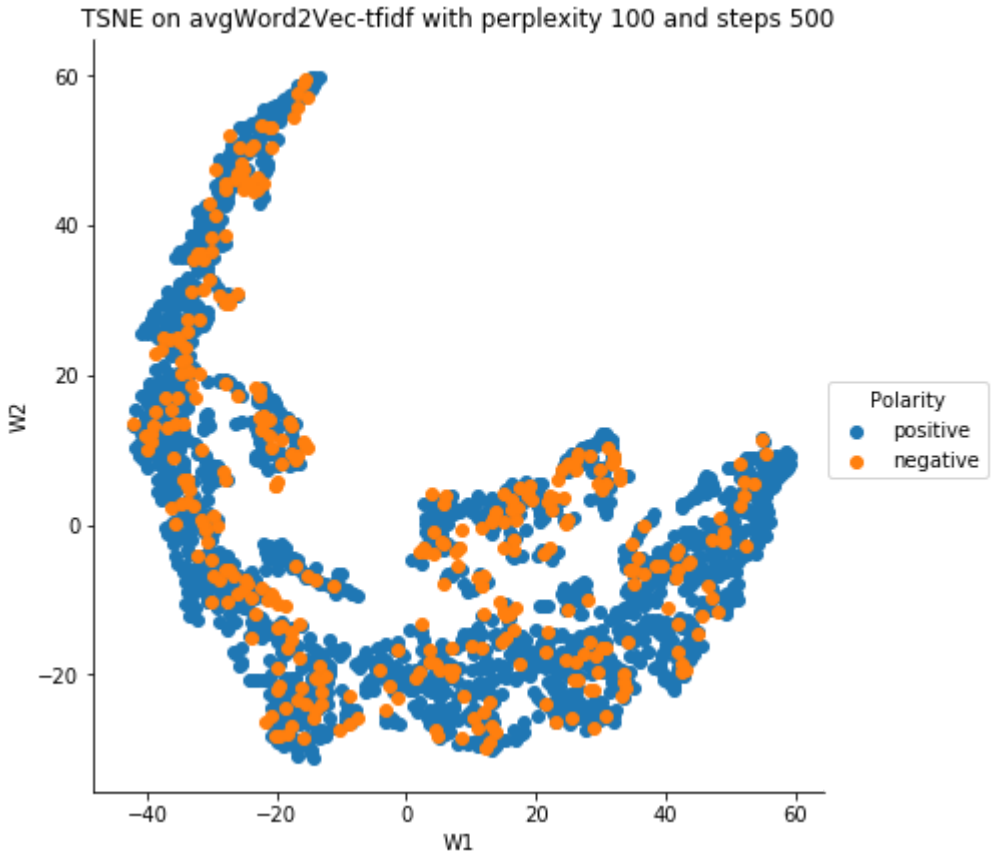
TSNE on avgWord2Vec also gives out a dispersed picture of the polarity. Not separable via a plane.

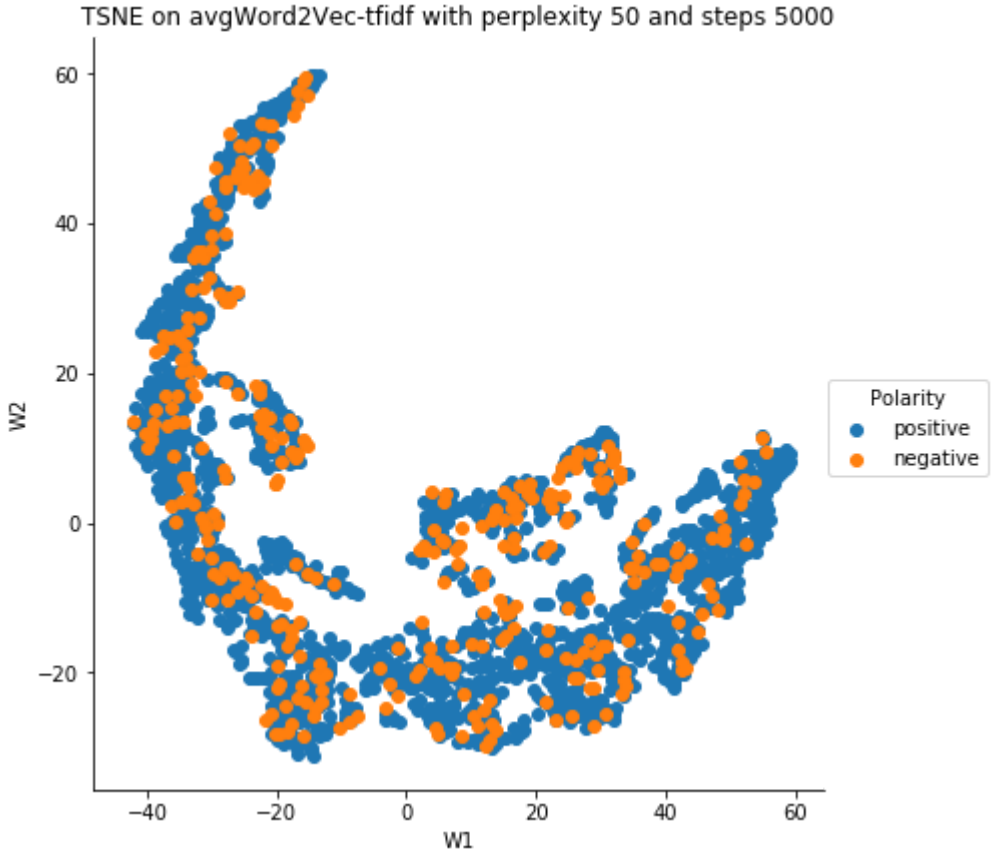
In [16]:
```
# TF-IDF weighted Word2Vec
tfidf_feat = tf_idf_vect.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val = t
final_tf_idf = tf_idf_vect.fit_transform(clean_reviews['CleanedText'].values)
tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored in th
row=0;
for sent in list_of_sent: # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    weight_sum =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            # obtain the tf_idfidf of a word in a sentence/review
            tf_idf =final_tf_idf[row, tfidf_feat.index(word)]
            sent_vec += (vec * tf_idf)
            weight_sum += tf_idf
    if weight_sum != 0:
        sent_vec /= weight_sum
    tfidf_sent_vectors.append(sent_vec)
    row += 1

final_counts=tfidf_sent_vectors
```

In [17]:
```
#TSNE with perplexity 100 and steps 1000
model=TSNE(n_components=2,random_state=0)
tsne_data=model.fit_transform(final_counts)
import numpy as np
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on avgWord2Vec-tfidf with perplexity 100 and steps 1000")
plt.show()


#TSNE with perplexity 30 and steps 1000
model=TSNE(n_components=2,random_state=0,perplexity=30)
tsne_data=model.fit_transform(final_counts)
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on avgWord2Vec-tfidf with perplexity 30 and steps 1000")
plt.show()

#TSNE with perplexity 50 and steps 5000
model=TSNE(n_components=2,random_state=0,perplexity=50,n_iter=5000)
tsne_data=model.fit_transform(final_counts)
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on avgWord2Vec-tfidf with perplexity 100 and steps 500")
plt.show()

#TSNE with perplexity 50 and steps 5000
model=TSNE(n_components=2,random_state=0,perplexity=50,n_iter=5000)
tsne_data=model.fit_transform(final_counts)
tsne_data=np.vstack((tsne_data.T,score.T)).T
tsne_data.shape
tsne_df=pd.DataFrame(data=tsne_data,columns=("W1","W2","Polarity"))
sns.FacetGrid(tsne_df,hue="Polarity",size=6).map(plt.scatter,"W1","W2").add_legen
plt.title("TSNE on avgWord2Vec-tfidf with perplexity 50 and steps 5000")
plt.show()
```


TSNE on avgWord2Vec-tfidf with perplexity 100 and steps 1000
W1
W2
Polarity
positive
negative


TSNE on avgWord2Vec-tfidf with perplexity 30 and steps 1000
W1
W2
Polarity
positive
negative





TSNE with different perplexity and steps on avgWord2Vec-tfidf also is not helpful. The figures do not allow separation of polarity of reviews using a linear figure.

Conclusion: We tried TSNE using different parameters (perplexity and steps) on different models namely Bag of Words, Tf-Idf, avgWord2Vec and avgWord2Vec-tfidf. The plots do not allow a separation of positive and negative reviews. The scatter plots are mostly dispersed with the both polarity of reviews spread uniformly. No plane can be drawn geometrically to separate the reviews and get model devised fot that.

###