Text sentiment classifier proposal

Genuine challenge S4-AI | Luuk de Kinderen

“Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor.”

Project goal

Several sites allow you to post a comment or opinion about a product, place, or event. Think of Facebook, Instagram, Youtube, or Reddit. However, on most of these sites, you can not see quickly the sentiment of these comments. We want to know if these comments are written with a positive or negative mindset.

For many people or companies, it is useful to know what percentage of the reactions are positive or negative. You can't always tell this by looking at a Like system.

Main question

The main question you should be able to answer with the help of this project is; *What is the most effective way to classify text sentiment?*

Project plan

To achieve this goal, I am going to use reviews. The advantage of reviews is that they are usually already labeled. For example, you can directly translate a star rating to positive or negative. And many review datasets are already labeled as positive and negative.

Context understanding

Review

A review is an evaluation of a publication, service, or company such as a movie, video game, musical composition, book; a piece of hardware like a car, home appliance, or computer; or an event or performance, such as a live music concert, play, musical theater show, dance show, or art exhibition.
(Wikipedia, 2021)

User Review

A user review refers to a review written by a user or consumer of a product or a service based on her experience as a user of the reviewed product. Popular sources for consumer reviews are e-commerce sites like Amazon.com, Zappos or lately in the Yoga field for schools such as Banjaara Yoga and Ayurveda, and social media sites like TripAdvisor and Yelp. E-commerce sites often have consumer reviews for products and sellers separately. Usually, consumer reviews are in the form of several lines of texts accompanied by a numerical rating. This text is meant to aid in shopping decision of a prospective buyer. A consumer review of a product usually comments on how well the product measures up to expectations based on the specifications provided by the manufacturer or seller. It talks about performance, reliability, quality defects, if any, and value for money. Consumer review, also called 'word of mouth' and 'user generated content' differs from 'marketer generated content' in its evaluation from consumer or user point of view. Often it includes comparative evaluations against competing products. Observations are factual as well as subjective in nature. Consumer review of sellers usually comment on service experienced, and dependability or trustworthiness of the seller. Usually, it comments on factors such as timeliness of delivery, packaging, and correctness of delivered items, shipping charges, return services against promises made, and so on.

Consumer reviews online have become a major factor in business reputation and brand image due to the popularity of TripAdvisor, Yelp, and online review websites. A negative review can damage the reputation of a business and this has created a new industry of reputation management where companies attempt to remove or hide bad reviews so that more favourable content is found when potential customers do research.
(Wikipedia, 2021)

Bought review

A bought review is the system where the creator (usually a company) of a new product pays a reviewer to review his new product. Primarily used in the car, movie, and game industry this system creates a kind of undercover advertising. Bought reviews tend to be biased due to the informative value of reviews. In some cases, a bought review may be independent, if the person that is hired to do the review has a strong reputation for independence and integrity. Even if a "bought review" from a respected critic is actually independent, the perception of potential bias will remain, due to the financial relationship between the

company and the critic.
(Wikipedia, 2021)

Positive or negative sentiment

What exactly makes text positive or negative? There are very many components that determine the sentiment of text. But by looking at certain keywords, you can most easily guess the sentiment of the text. Think about words like "good" or "bad." These words by themselves are literal translations of sentiment.

Negative

You can often use other components to guess the sentiment of the text. Negative text, for example, is more often written in the past tense. When people are negative about something they have more often than not already put it behind them. So it is no longer a problem of the present. For example; “This product was very bad” is written in the past tense. While; “This product is very good” is written in present tense.
Negative reviews also use contradictions more often. Signal words such as "but" and "or" are much more common in negative text than in positive text. For example; “It was okay, but it could have been much better.” Is clearly a negative text.

Positive

It is noticeable that there are many more keywords for positive text than negative text. Think "lovely", "great", "excellent" etc. These are all words that are a direct translation of a positive atmosphere/setting.
In addition, you can recognize positive text by one more thing. When someone is positive about something he/she also wants to tell more about it. So when people express themselves positively about something or someone they use more words on average. In addition, when someone expresses themselves positively, more positive things are mentioned. The keyword "and" is also more often found in positive text.

Potential Impact assessment

Impact on society

- *What is the challenge at hand? What problem (what 'pain') does this technology want to solve*
  Using the technology a user can predict the sentiment of a piece of text without even reading the text.
- *Can you motivate why you are sure that this technology is solving the right problem?*
  It can be very useful for different people, to know within a few seconds whether the text is positive or negative. If you quickly want to know the average opinion about something or someone. Then this tool is perfect to use.
- *In what way is this technology contributing to a world you want to live in?*
  In this world, we want to move fast. Getting results as fast as possible is important to compete with others. This technology helps the world to move even faster. And that alone really excites me.
- *Now that you have thought hard about the impact of this technology on society, what improvements would you like to make? List them below.*
    1. Combine this technology with a web scraper. That way manual input of multiple texts becomes obsolete. And getting results will be even faster.
    2. Adding a percentage calculator while using more inputs. If you have multiple inputs of different pieces of text. Knowing what percentage of the texts is classified as positive or negative will be really interesting to add.
    3. Showing the classification certainty while making conclusions. This certainty can be interesting for the users.

Stakeholders

- *What are the main users/target groups/stakeholders for this technology?*
    - **Companies that produce/deliver products/services**
      By having a clear overview of the average overall experience of a product/service early on, companies can adapt quickly if they know in time that the average opinion of their own product/service has changed. In addition, it can be helpful to know the average opinion about similar products/services of competing companies. In fact, a company can compete better when they know the overall opinion of their competitor.

      This stakeholder is a large group, but a very important one. If this technology is created, it will have a big impact on the market they are in. So this stakeholder is definitely one to take into account.

    - **Companies that sell products/services**
      If you sell multiple products/services, it is very helpful to know which of these are more popular with customers than others. As a company, you can then better advertise these products/services.

    - **Consumers of products/services**
      Of course, if using this technology changes the development and promotion of products/services, it also affects consumers. The choice between products/services will change.

    - **Pollsters**
      This technology will be a revolution for pollsters. Not taking into account this stakeholder is unavoidable. It will be a big change because much of their work will be automated and this will simplify their work. However, this does not mean that it will be positive for everyone, as it may mean that jobs will disappear.

    - **News channels**
      When a news channel wants to convey an insight of opinions to the viewer. The help of pollsters is often enlisted. With the help of this technology, this will become unnecessary in some cases. Channels will be able to come up with results more quickly themselves.

    - **Politicians**
      People's opinion about an idea or political proposal is very important for politicians to know. If the majority of all people are negative about a proposal, it can have dramatic consequences for a politician and his/her entire party. Of course, this also works the other way around. If a politician or party comes up with an idea that is well received by everyone, this may mean that a party will grow in voters.

      When the use of this technology becomes mainstream. It would mean that politicians would handle the opinion of voters differently. Everyone can see

the average opinion of voters and this can also be used as an argument in a debate.

This stakeholder is certainly important to take into account because it has a big impact on their way of working. And the work of politicians, of course, has an effect on everyone.

- *Now that you have thought hard about all stakeholders, what improvements would you like to make? List them below.*
    1. Adding a warning that the classification is a prediction would be a good addition. By adding this warning you prevent users from using the results as a fact.
    2. It might be a good addition that when a prediction is published, it should include a source. Such as: "According to the opinion forecaster". This will prevent the technology from being used as a fact.

Privacy

- *Does this technology register personal data? If yes, what personal data?*
  no, this technology only uses personal data to come to a prediction. This data is not stored. Whether the data is actually personal is also a matter of debate. It only uses a piece of text and no name is attached to it.

- *Do you think this technology invades someones privacy? If yes, in what way?*
  When this technology is used by someone. It may be that another person's opinion is used to make a conclusion. This could be interpreted as a privacy violation. But this was also possible without this technology. The comments of others are always visible and conclusions can be drawn from them. The only thing that has changed is that this technology makes this easier and faster.

- *Do you think this technology is compliant with prevailing privacy and data protection law and can you motivate why?*
  I don't think this technology violates the current privacy law. All the data being used is already public for all to see. Drawing a conclusion from this is not illegal and is currently already often done by people.

- *In which way can you imagine a future impact of the collection of personal data?*
  This technology does not store any data. But in general, it is very important that personal data is not leaked and does not end up with people for whom it is not intended.

- *Now that you have thought hard about privacy and data protection, what improvements would you like to make? List them below.*
  This technology does not store any data. So I have not come up with any further additions to the technology.

Transparency

- *(How) is explained to the users how a technology works and how the businessmodel works?*
  If this technology can actually be used through a website. I can imagine there being an 'about' page explaining how the technology works. In addition, the notebook used to develop the technology is publicly available on github. People with knowledge about machine learning can then see for themselves how this technology was developed.

- *Is it possible to file a complaint or ask questions/get answers about this technology?*
  If this technology can actually be used through a website. Then there will also be a 'contact' page. Otherwise, people can look up my contact information on github.

- *Is the technology (company) clear about possible negative consequences or shortcomings of this technology?*
  There will be a clear warning that this technology can only be used for predictions. And that the prediction may not be accurate.

- *Now that you have thought hard about the transparency of this technology, what improvements would you like to make? List them below.*
  1. A terms of use stating that the technology should not be used for negative objectives would be a good addition.

Inclusivity

- *Will everyone have access to this technology?*
  Yes, if you have a computer you basically have access to this technology.

- *Does this technology have a built in bias?*
  no, as far as can be estimated in advance, this technique has no biases built in.

- *Does this technology make automatic decisions and how do you account for them?*
  Yes, this technology makes automatic decisions on its own. This is handled by giving a clear warning that a prediction should not be taken as truth.

- *Is everyone benefitting from this technology or only a small group? Do you see this as a problem? Why/why not?*
  Only users of the technology will benefit. I don't see this as a problem, because I don't expect non-users to be disadvantaged by this.

- *Does the team that creates the technology represents the diversity of our society?*
  No, this project is made by one person. This automatically means that an individual does not reflect the diversity of our society. But I don't think this is not a problem.

- *Now that you have thought hard about the inclusivity of this technology, what improvements would you like to make? List them below.*
  There are no improvements I would like to apply to this technology in terms of inclusivity.

Sustainability

- *In what way is the direct and indirect energy use of this technology taken into account?*
  I expect that the energy consumption of this technology is so low that it is not taken into account at the moment.

- *Do you think alternative materials could have been considered in this technology?*
  Yes, this technology is developed on a computer with the programming language python. According to an article on thenewstack by David Cassel, python is the second last energy-efficient programming language from that study. So a different programming language will probably make the total energy consumption lower.

- *Do you think the lifespan of this technology is realistic?*
  Yes, I think this technology can be regularly improved or extended by adding multiple new reviews.

- *What is the hidden impact of this technology in the whole chain?*
  I don't think there is a hidden impact of this technology.

- *Now that you have thought hard about the sustainability of this technology, what improvements would you like to make? List them below.*
  Using a different programming language probably would be better for the environment. But most of this semester is based on using python. Because of that, I will not change the programming language I work with.

Hateful and criminal actors

- *In which way can this technology be used to break the law or avoid the consequences of breaking the law?*
  This technology cannot be used directly to break the law. However, this technology can be used to develop sympathy. When someone has done something criminal and he/she can show with a (fake) report, developed with this technology, that the average opinion is positive. So then he/she can create a (fake) sense of sympathy.

- *Can you imagine this technology being used to cross personal- or societal boundaries?*
  In some extreme cases, this might be possible. People can quickly use average opinion as an argument. This could cross certain boundaries. But I don't expect this to have dire consequences.

- *Can this technology be used against certain (ethnic) groups or (social) classes?*
  Yes, when a classification is done using only the opinion of certain (etic) groups or (social) classes, then this can also be used against them. But to do this, a selection in the input has to be done beforehand. Someone is then deliberately doing this.

- *In which way can bad actors use this technology to pit certain groups against each other? These groups can be, but are not constrained to, ethnic, social, political or religious groups.*
  Someone can give extra power to the opinion of a group by making this an average opinion. Using the average opinion may make it seem more likely that a whole group has that opinion. As a result, there may be an even greater difference between the 2 groups when the average opinion is used.

- *How could bad actors use this technology to subvert or attack the truth?*
  People might use a fake result to make an opinion seem different than it actually is.

- *Now that you have thought hard about how bad actors can impact this technology, what improvements would you like to make? List them below.*
  1. Have some kind of warning that this technology should not be used for bad purposes.
  2. create awareness that the results of this technology cannot always be assumed to be true.

Human values

- *How does your technology affect the identity of the user(s)?*
  By using this technology you can be seen as a geek. Apart from that, I don't think your identity changes after using this technology.

- *How does the technology influence the user(s) autonomy?*
  It perhaps changes a user's autonomy to have their own judgment. Now people themselves often have a judgment whether someone meant something positive or negative. But with the use of this technology, this perspective might disappear.

- *What is the effect of the technology on the health and/or wellbeing of the user(s)?*
  There is no effect on the health and/or wellbeing of the user.

- *Now that you have thought hard about the impact of your technology on human values, what improvements would you like to make? List them below.*
  There are no improvements I would like to apply to this technology in terms of Human values.

Future

- *What could possibly happen with this technology in the future?*
  The technology can become much more precise. Whereby the result can be taken more and more seriously. To a point that people start believing in it. In the same way that people also take a weather forecast for truth, and get angry when it is not true.

- *Sketch a or some future scenario (s) (20-50 years up front) with the help of storytelling. Start with at least one utopian scenario.*
  In the ideal world, this technology is used to make life a lot easier. Quickly knowing the general opinion of others can be positively applied in many different ways. Products are improved and produced faster. Politicians can act faster and do what the average citizen wants. And news that incorporates an opinion is made even faster.

- *Sketch a or some future scenario (s) (20-50 years up front) with the help of storytelling. Start with at least one dystopian scenario.*
  In a dystopian scenario, this technique is used only for evil purposes. Groups of people are more quickly pigeonholed by the expression of an opinion. People need to think more carefully about how and if they express their opinions. The political system may collapse because all decisions are based on people's average opinions. All opinion pollsters will be out of work and this technology will only do further damage.

- *Would you like to live in one of this scenario's? Why? Why not?*
  The first scenario seems pleasant to live in and also the most predictable. In general, there are still more good people than bad, and I don't expect this technology to change that.

- *What happens if your technology (which you have thought of as ethically well-considered) is bought or taken over by another party?*
  I don't think another party will change much. They might remove certain warnings or terms of use, but I expect most people are objective enough at the moment and can think for themselves about the consequences of making a classification.

Potential Impact assessment conclusion

Properly informing what the consequences may be is very important. If this technology would actually be available as a tool/website, then it is important to inform users properly.

It is also wise to add a term of use that makes it mandatory to name the tool when it is used in a report. Like “According to the Text sentiment classifier, the average result is …” This will prevent the average opinion from being seen as truth.

These terms of use will also state that it is mandatory to mention that results are a prediction and cannot be used as a fact or argument. The tool is only made to quickly display general information, not for drawing conclusions.

Datasets

I am going to use 3 datasets that are as different as possible. To keep the training input as variable as possible, I searched for 3 different topics with different structures.

Movies

This is a dataset consisting of 1000 positive and 1000 negative reviews of movies. Unfortunately, it is not known which films the reviews are about. What is known is that they were scraped from imdb.com.

Link: https://www.kaggle.com/nltkdata/movie-review

Each row contains a single sentence of a review. This results in very short review texts.

Hotels

This dataset contains 515,000 customer reviews and scoring of 1493 luxury hotels across Europe. The data was scraped from booking.com. Each row consists of very much review and hotel information. This is very interesting for further analysis!

There is one thing unique about this dataset compared to the others. Each reviewer had to fill in a text field for both positive and for negative review text.

Link: https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe

Because some visitors were very satisfied or unsatisfied, they did not leave a negative or positive review. Or they filled this in as; "nothing positive to report". These kinds of values need to be filtered out. This is something that requires extra attention.

Amazon products

There are very many different Amazon review datasets available. I chose the "Toys and Games" category because it doesn't have much to do with the other topics.

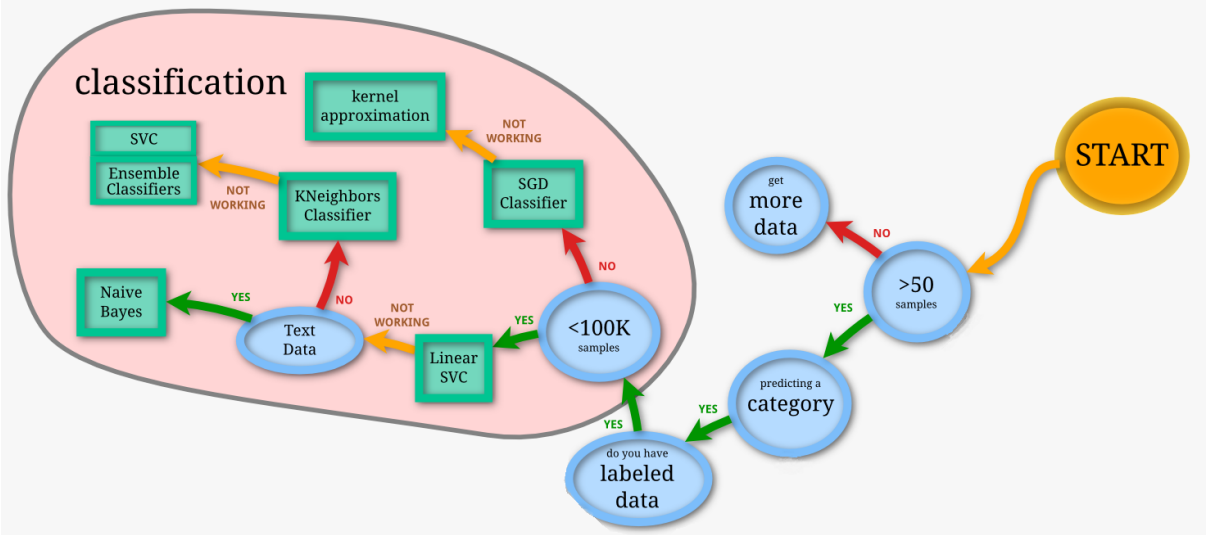
The toys and games dataset consists of 1,828,971 unique reviews, across 78,772 unique products. The data was scraped directly from amazon.com.

Link: https://nijianmo.github.io/amazon/index.html

This dataset has not yet been directly labeled positive or negative. But consists of a star rating from 1 to 5 stars. So we need to find a good and fair way to convert this rating to labels.

Modelling

There are very many ways you can classify. A good start is to start by looking at the cheat sheet of Scikit-learn.org



When I look at my options I come to two things: Linear SVC and Naive bayes.

Linear SVC

Linear svc is a straightforward way of classifying. I am familiar and comfortable with it. I do want to look further into other kernels beyond linear. There are many more options and I would like to research and test with them.

Naive bayes

I have very high hopes for Naive bayes. After all, it is known to work very well with text data. Exactly what this project is about! There are, as with SVC, many different versions of Naive Bayes. To find the best model, I would like to try them all and compare the differences.

Evaluation

At this point, I have learned a lot about the subject. I know what text sentiment means and the difference between positive and negative text. Also, I know what a review is and the different types of reviews.
A clear plan has been made for me and I know what I need to achieve it. However, we have yet to determine when the goal has been achieved.

Domain knowledge verification

How do we know when the model is working well? When the model classifies text sentiment, when will we know if it has done this correctly? This is hard to say because the model is trained with labled data. This means that the model trains based on the labels in the training data. However, we cannot be sure in advance that these labels are correct. Assuming that these labels are correct, we can make a comparison between the predicted data and test data. I can assume that the model works well when there is little difference between the predicted data and the test data.

Results

We can be satisfied when the model is between 70% and 80% accurate. However, it is important that the difference between the number of false positives and false negatives is as low as possible. This is important because we only have 2 lables.

Deployment

To make the model available to everyone we use anvil. This is a tool that allows you to create beautiful drag and drop UIs and share your local python functions with anyone via a websocket connection.

Literature list

- (2021, 16 March). Review.
  Consulted from https://en.wikipedia.org/wiki/Review
- (2018, 20 May) Which Programming Languages Use the Least Electricity, by David Cassel
  Consulted from
  https://thenewstack.io/which-programming-languages-use-the-least-electricity/

EDA Amazon

Project

Several sites allow you to post a comment or opinion about a product, place, or event. Think of Facebook, Instagram, Youtube, or Reddit. However, on not all of these sites, you can see at a glance how many of these comments are written with a positive or negative mindset.

For many people or companies, it is useful to know what percentage of the reactions are positive or negative. You can't always tell this by looking at a Like system. The main question you should be able to answer with the help of this project is; Is a piece of text positive or negative?

Amazon Dataset

This is a dataset with all reviews coming from the site Amazon. source:
https://nijianmo.github.io/amazon/index.html (https://nijianmo.github.io/amazon/index.html)

Document goal

In this paper, I want to gain more insight into the amazon dataset. And estimate if it is useful for achieving my project goal.

First We will start with importing libraries and doing some styling

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

#set decimal point
pd.options.display.float_format = "{:.5f}".format
```

In [2]:
```
%%html
<style>
table {float:left}
</style>
```

To load the data as pandas dataframe 2 functions are needed. Those are fortunately found on the source page.

In [3]:
```
#import libraries
import gzip
import json

#functions copied from source
def parse(path):
  g = gzip.open(path, 'rb')
  for l in g:
    yield json.loads(l)

def getDF(path):
  i = 0
  df = {}
  for d in parse(path):
    df[i] = d
    i += 1
  return pd.DataFrame.from_dict(df, orient='index')

#use functions to create dataframe
df = getDF('data/raw/Toys_and_Games_5.json.gz')
```

Now that we have a data frame let's start looking at the data

In [4]:
```
df.head()
```

Out[4]:

	overall	vote	verified	reviewTime	reviewerID	asin	style	reviewerName
0	5.00000	3	True	10 6, 2013	A2LSCFZM2FBZK7	0486427706	{'Format:': ' Paperback'}	Ginger
1	5.00000	9	True	08 9, 2013	A3IXP5VS847GE5	0486427706	{'Format:': ' Paperback'}	Dragonflies & Autumn Leaves
2	5.00000	NaN	True	04 5, 2016	A1274GG1EB2JLJ	0486427706	{'Format:': ' Paperback'}	barbara ann
3	5.00000	3	True	02 13, 2016	A30X5EGBYAZQQK	0486427706	{'Format:': ' Paperback'}	Samantha
4	5.00000	NaN	True	12 10, 2015	A3U6UNXLAUY6ZV	0486427706	{'Format:': ' Paperback'}	CP in Texas

I can already see some missing values. Let's see how many there are

In [5]:
```
1 df.isnull().sum()
```

Out[5]:
```
overall              0
vote           1620926
verified             0
reviewTime           0
reviewerID           0
asin                 0
style          1305477
reviewerName       116
reviewText        1175
summary            372
unixReviewTime       0
image          1787588
dtype: int64
```

And how much of a percentage is that?

In [6]:
```
1 df.isnull().sum().apply(lambda x: 100/ len(df) * x)
```

Out[6]:
```
overall          0.00000
vote            88.62502
verified         0.00000
reviewTime       0.00000
reviewerID       0.00000
asin             0.00000
style           71.37768
reviewerName     0.00634
reviewText       0.06424
summary          0.02034
unixReviewTime   0.00000
image           97.73736
dtype: float64
```

In `image`, `style` and `vote`, a lot of values are missing. I am most interested in the `reviewText` so probably this is not a problem. But what kind of fields are these exactly? Let's see how it is described on the source.

Column information

Column name	Description	Data Type
overall	the overal score a product	float
vote	helpfull votes of review	float
verified	reviewer verified state	bool
reviewTime	date of the review	datetime
reviewerID	The ID of the reviewer	object
asin	The ID of the product	object
reviewerName	The name of the reviewer	object
summary	The summary of the review	object

Column name	Description	Data Type
unixReviewTime	date of the review in unix format	int
style	A summary of product specifications, e.g., "Color" or "Size"	object
image	Images the reviewer uploaded after recieving the product	object

How is our data loaded?

In [7]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1828971 entries, 0 to 1828970
Data columns (total 12 columns):
 #   Column          Dtype  
---  ------          -----  
 0   overall         float64
 1   vote            object 
 2   verified        bool   
 3   reviewTime      object 
 4   reviewerID      object 
 5   asin            object 
 6   style           object 
 7   reviewerName    object 
 8   reviewText      object 
 9   summary         object 
 10  unixReviewTime  int64  
 11  image           object 
dtypes: bool(1), float64(1), int64(1), object(9)
memory usage: 169.2+ MB
```

Data cleaning

some columns are not of the type indicated in the source. We are going to modify this.

Vote is of type `object` , according to the dataset page this really should be a numeric type. After looking into it I found out some values are writen as `1,000` in stead of `1000` So lets remove all the comma's `,` and make it numeric.

Additionally, currently there are two columns that refer to the time when the review was posted. We are going to replace these with a column of type datetime.

In [8]:
```
#clean out ','
df['vote'] = df['vote'].str.replace(",", "")
#change to numeric type
df['vote'] = pd.to_numeric(df['vote'])

#change to datetime type
df['unixReviewTime'] = pd.to_datetime(df['unixReviewTime'],unit='s')
#remove old review time column
del df['reviewTime']
#rename column so 'unix' is removed from column name
df = df.rename(columns = {'unixReviewTime': 'reviewTime'}, inplace = False)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1828971 entries, 0 to 1828970
Data columns (total 11 columns):
 #   Column        Dtype         
---  ------        -----         
 0   overall       float64       
 1   vote          float64       
 2   verified      bool          
 3   reviewerID    object        
 4   asin          object        
 5   style         object        
 6   reviewerName  object        
 7   reviewText    object        
 8   summary       object        
 9   reviewTime    datetime64[ns]
 10  image         object        
dtypes: bool(1), datetime64[ns](1), float64(2), object(7)
memory usage: 155.2+ MB
```

Data ledger

Where is our data comming from?

Concerning fact	Amazon Review Data
Source	https://nijianmo.github.io/amazon/index.html (https://nijianmo.github.io/amazon/index.html)
Origin of the data	Amazon
rate of refreshing	NaN
date of the data	Oct 2018
Period the data concerns	May 1996 - Oct 2018
Size of data	**Size**: Approx. 231 MB, **Rows**: 1828971, **Columns**: 12

Let's look at some average values. And some deeper insights

In [9]:
```
1 df.describe(include='all', datetime_is_numeric=True)
```

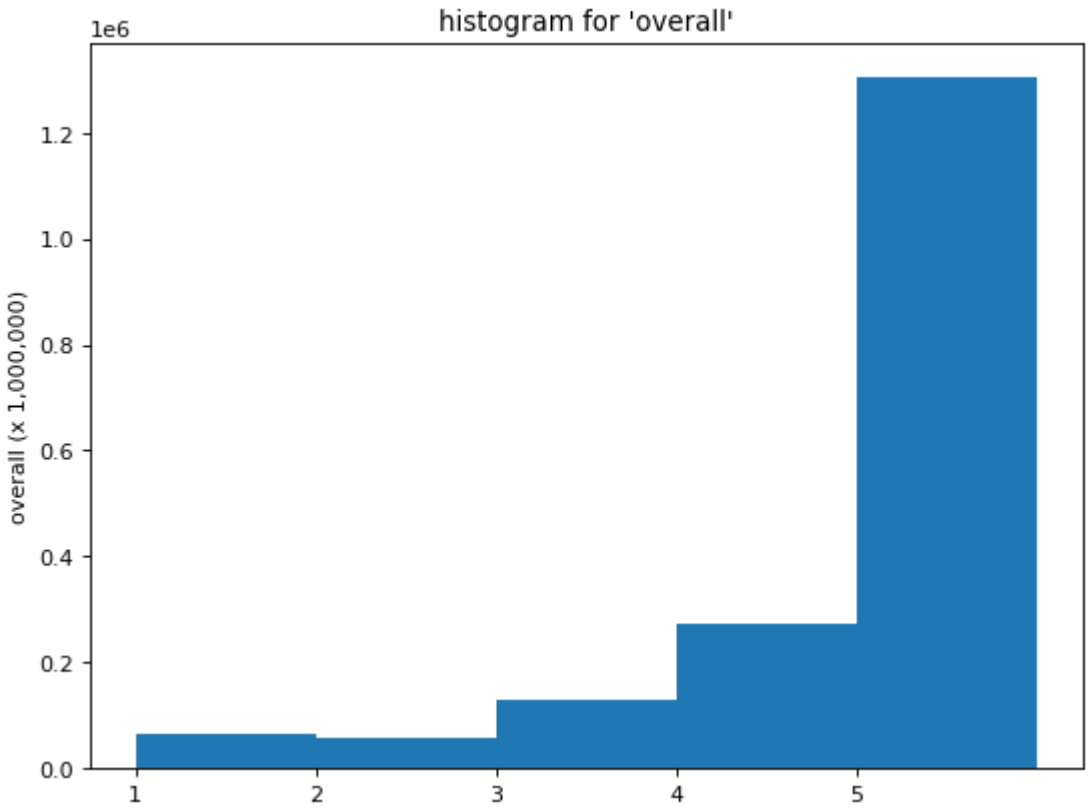
Out[9]:

	overall	vote	verified	reviewerID	asin	style	review
count	1828971.00000	208045.00000	1828971	1828971	1828971	523494	
unique	nan	nan	2	208180	78772	12983	
top	nan	nan	True	AJGU56YG8G1DQ	B000YDDF6O	{'Product Packaging:': ' Standard Packaging'}	C
freq	nan	nan	1599831	812	4887	67788	
mean	4.47415	7.57850	NaN	NaN	NaN	NaN	
min	1.00000	2.00000	NaN	NaN	NaN	NaN	
25%	4.00000	2.00000	NaN	NaN	NaN	NaN	
50%	5.00000	3.00000	NaN	NaN	NaN	NaN	
75%	5.00000	6.00000	NaN	NaN	NaN	NaN	
max	5.00000	1973.00000	NaN	NaN	NaN	NaN	
std	0.99968	24.08011	NaN	NaN	NaN	NaN	

Notable is that the mean of `overall`, is `4.474...` .And the maximum is `5`. So probably this dataset consists of a lot more positive reviews than negative reviews. Also, if you look at the most used word in the `reviewText`: `great`, you can predict that there are more positive reviews than negative reviews. A histogram of `overall` can probably clearly show the distribution of ratings.

In [10]:

```
#make figure bigger
figure(figsize=(8, 6), dpi=80)

#get amount of bars
n_bars = int(df['overall'].max())

#set number of bars
plt.xticks(range(n_bars+1))
#make histogram
plt.hist(df['overall'],bins=5, range=(1,n_bars+1))
#add text
plt.ylabel('overall (x 1,000,000)')
plt.title("histogram for 'overall'")
# show
plt.show()
```



As I expected, there are many more reviews with a high score instead of a low one.

Add a Sentiment colum

To make classification easier later on, it is useful and convenient to add a new column. This will be called `sentiment`, and has the following values:

if `overall` Is `3` the sentiment is `NEUTRAL`
if `overall` Is `4` or `5` the sentiment is `POSITIVE`
if `overall` Is `1` or `2` the sentiment is `NEGATIVE`

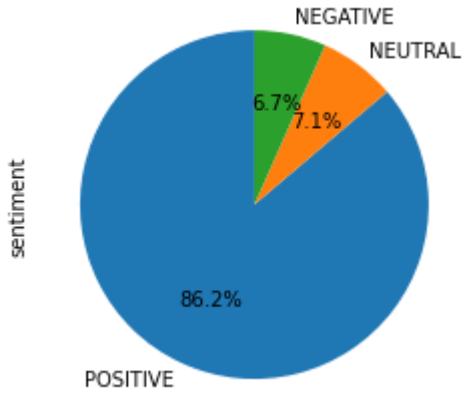
In [11]:
```
def getSentiment(overall):
    sentiment = 'NEUTRAL'
    if overall > 3:
        sentiment = 'POSITIVE'
    if overall < 3:
        sentiment = 'NEGATIVE'
    return sentiment

df['sentiment'] = df['overall'].apply(getSentiment)

df.tail()
```

Out[11]:

	overall	vote	verified	reviewerID	asin	style	reviewerName	revie
1828966	5.00000	nan	True	A3RRY2TCWM3MWU	B01HJ4GZIU	NaN	ash	This hug purcl this a
1828967	1.00000	nan	True	A2ZRCFLXI8J39O	B01HJ4GZIU	NaN	Laura	Ours work 3 w out
1828968	5.00000	nan	True	A2IIP6L4PRAXVN	B01HJ8SCA0	NaN	JM	S play Ev gra cant
1828969	5.00000	nan	True	ACJH8FKNLSXIQ	B01HJDFWDK	{'Color:': ' 9.6v Battery pack'}	Steve Chen	These we are cl from
1828970	5.00000	nan	True	A2KVWRRMFOYCW	B01HJDFWDK	{'Color:': ' 6v Battery pack'}	Rocco245	b Use i e

How is Sentiment distributed?

In [12]:
```
fig, ax = plt.subplots()
df['sentiment'].value_counts().plot(ax=ax, kind='pie', autopct='%1.1f%%', st
plt.show()
```



This pie graph really shows that the sentiment colum is not evenly distributed.

Data filtering

For this project, we are only interested in two things:

1. A piece of text
2. The sentiment

So let's filter out all other data.

Other datasets have no `NEUTRAL` values. So we are going to filter these out as well.

In [13]:
```
#filter on text and sentiment
df = df.filter(['reviewText', 'sentiment'], axis=1)

#filter out NEUTAL
df = df[df['sentiment'] != 'NEUTRAL']

#shuffle
df = df.sample(frac=1, random_state=42)
#drop index
df = df.reset_index(drop=True)

df
```

Out[13]:

1698454 rows × 2 columns

	reviewText	sentiment
0	Over all the figure is great. The only downfal...	POSITIVE
1	Great puzzle- not too hard for the 3 year old.	POSITIVE
2	This is a GREAT toy!	POSITIVE
3	almost as good as regular Fluxx and Star Fluxx.	POSITIVE
4	daughter loves it	POSITIVE
...	...	...
1698449	Great, wooden, life-like toys!	POSITIVE
1698450	great service & product	POSITIVE
1698451	My 12 month old son loves putting things in ho...	NEGATIVE
1698452	These are high-quality beads. Cute and vibrant...	POSITIVE
1698453	Decent quality. Good size. Kids jump on the d...	POSITIVE

Conclusion

I now have a better understanding of the data in the dataset. I have filtered for what is useful and added a sentiment column. I feel confident that I can use this data in the next step.

Save DF

We are done with filtering and exploring the data of this dataset. Now we save it so it can be used in another file.

In [14]:
```
#save dataframe as csv to use it in another file
df.to_csv(r'data/filtered/amazon_filtered.csv', index = False)
```

EDA Hotel

Project

Several sites allow you to post a comment or opinion about a product, place, or event. Think of Facebook, Instagram, Youtube, or Reddit. However, on not all of these sites, you can see at a glance how many of these comments are written with a positive or negative mindset.

For many people or companies, it is useful to know what percentage of the reactions are positive or negative. You can't always tell this by looking at a Like system. The main question you should be able to answer with the help of this project is; Is a piece of text positive or negative?

Hotel Dataset

This datasets contains data from a lot of hotels. Each reviewer left a positive and a negative review. The data was scraped from https://Booking.com (https://Booking.com). It comes from kaggle.

source: https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe (https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe)

Document goal

In this paper, I want to gain more insight into the Hotel dataset. And estimate if it is useful for achieving my project goal.

First We will start with importing libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

#set decimal point
pd.options.display.float_format = "{:.5f}".format
```

Let's start by loading the data set

In [2]:
```
df = pd.read_csv('data/raw/Hotel_Reviews.csv')
```

Now that we have a data frame let's start looking at the data

In [3]:
```
1 df
```

Out[3]:

	Hotel_Address	Additional_Number_of_Scoring	Review_Date	Average_Score	Hotel_Nar
0	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	8/3/2017	7.70000	Hotel Are
1	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	8/3/2017	7.70000	Hotel Are
2	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/31/2017	7.70000	Hotel Are
3	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/31/2017	7.70000	Hotel Are
4	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/24/2017	7.70000	Hotel Are
...	...	...	...	...	
515733	Wurzbachgasse 21 15 Rudolfsheim F nfhaus 1150 ...	168	8/30/2015	8.10000	Atlan Hotel Vien
515734	Wurzbachgasse 21 15 Rudolfsheim F nfhaus 1150 ...	168	8/22/2015	8.10000	Atlan Hotel Vien
515735	Wurzbachgasse 21 15 Rudolfsheim F nfhaus 1150 ...	168	8/19/2015	8.10000	Atlan Hotel Vien
515736	Wurzbachgasse 21 15 Rudolfsheim F nfhaus 1150 ...	168	8/17/2015	8.10000	Atlan Hotel Vien

	Hotel_Address	Additional_Number_of_Scoring	Review_Date	Average_Score	Hotel_Nar
515737	Wurzbachgasse 21 15 Rudolfsheim F nfhaus 1150 ...	168	8/9/2015	8.10000	Atlan Hotel Vien

515738 rows × 17 columns

Data ledger

Where is our data comming from?

Concerning fact	Hotel Reviews Data in Europe
Source	https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe (https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe)
Origin of the data	The data was scraped from Booking.com.
rate of refreshing	NaN
date of the data	2017-03-08
Period the data concerns	2015-08-04 - 2017-03-08
Size of data	**Size**: Approx. 227 MB, **Rows**: 515738 , **Columns**: 17

In [4]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 515738 entries, 0 to 515737
Data columns (total 17 columns):
 #   Column                                      Non-Null Count   Dtype  
---  ------                                      --------------   -----  
 0   Hotel_Address                               515738 non-null  object 
 1   Additional_Number_of_Scoring                515738 non-null  int64  
 2   Review_Date                                 515738 non-null  object 
 3   Average_Score                               515738 non-null  float64
 4   Hotel_Name                                  515738 non-null  object 
 5   Reviewer_Nationality                        515738 non-null  object 
 6   Negative_Review                             515738 non-null  object 
 7   Review_Total_Negative_Word_Counts           515738 non-null  int64  
 8   Total_Number_of_Reviews                     515738 non-null  int64  
 9   Positive_Review                             515738 non-null  object 
 10  Review_Total_Positive_Word_Counts           515738 non-null  int64  
 11  Total_Number_of_Reviews_Reviewer_Has_Given  515738 non-null  int64  
 12  Reviewer_Score                              515738 non-null  float64
 13  Tags                                        515738 non-null  object 
 14  days_since_review                           515738 non-null  object 
 15  lat                                         512470 non-null  float64
 16  lng                                         512470 non-null  float64
dtypes: float64(4), int64(5), object(8)
memory usage: 66.9+ MB
```

In [5]:
```
1 df.describe(include='all', datetime_is_numeric=True)
```

Out[5]:

	Hotel_Address	Additional_Number_of_Scoring	Review_Date	Average_Score	Hotel_Name	R
count	515738	515738.00000	515738	515738.00000	515738	
unique	1493	nan	731	nan	1492	
top	163 Marsh Wall Docklands Tower Hamlets London ...	nan	8/2/2017	nan	Britannia International Hotel Canary Wharf	
freq	4789	nan	2585	nan	4789	
mean	NaN	498.08184	NaN	8.39749	NaN	
std	NaN	500.53847	NaN	0.54805	NaN	
min	NaN	1.00000	NaN	5.20000	NaN	
25%	NaN	169.00000	NaN	8.10000	NaN	
50%	NaN	341.00000	NaN	8.40000	NaN	
75%	NaN	660.00000	NaN	8.80000	NaN	
max	NaN	2682.00000	NaN	9.80000	NaN	

Data cleaning

I found out that the word count provided by the dataframe is not correct.

In [6]:
```
#positive
print("pos sentence: ", df['Positive_Review'][5])
print("dataframe word count : ", df['Review_Total_Positive_Word_Counts'][5])
print("calculated word count: ", len(df['Positive_Review'][5].split()))

#negative
print("neg sentence: ", df['Negative_Review'][5])
print("dataframe word count : ", df['Review_Total_Negative_Word_Counts'][5])
print("calculated word count: ", len(df['Negative_Review'][5].split()))
```

```
pos sentence:   Good restaurant with modern design great chill out place Great
park nearby the hotel and awesome main stairs
dataframe word count :  20
calculated word count:  18
neg sentence:   Backyard of the hotel is total mess shouldn t happen in hotel w
ith 4 stars
dataframe word count :  17
calculated word count:  15
```

So I created a function that overrides the wordcount

In [7]:
```
def fixWordCount():
    df['Review_Total_Positive_Word_Counts'] = df['Positive_Review'].str.spli
    df['Review_Total_Negative_Word_Counts'] = df['Negative_Review'].str.spli

fixWordCount()
```

`Review_Date` is not of type datetime. Let's fix that

In [8]:
```
df['Review_Date'] = pd.to_datetime(df['Review_Date'])
```

In each row there is a field for positive and negative text. While filling out, many people have entered text such as "absolutely **nothing**" or "**nothing** at all" in the input field. We want to filter this out of the dataset, because "**nothing** at all" is not a good training text. We achieve this by filtering for the `keyword` **nothing** and then replacing the entire text with an empty string. But we only want to do this above a certain `word cap` because phrases like "the staff was very bad they did **nothing** at all and the food was very bad." do make good training sentences.

Furthermore, when people have not filled in anything it is replaced with **No Negative** or **No Positive**. We also want to replace this text with empty strings. Lets add those words to the `keywords` .

In [9]:
```
keywords = ['nothing' , 'Nothing', 'No Negative', 'No Positive']
wordcap = 15

df.loc[(df['Negative_Review'].str.contains('|'.join(keywords))) & (df['Revie
df.loc[(df['Positive_Review'].str.contains('|'.join(keywords))) & (df['Revie

```

Now we need to rerun the word count fixer because we deleted some reviews

In [10]:
```
fixWordCount()
```

Column information

Now that all the data has been cleaned up, we can create a column information table.

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 515738 entries, 0 to 515737
Data columns (total 17 columns):
 #   Column                                      Non-Null Count   Dtype
---  ------                                      --------------   -----
 0   Hotel_Address                               515738 non-null  object
 1   Additional_Number_of_Scoring                515738 non-null  int64
 2   Review_Date                                 515738 non-null  datetime64
[ns]
 3   Average_Score                               515738 non-null  float64
 4   Hotel_Name                                  515738 non-null  object
 5   Reviewer_Nationality                        515738 non-null  object
 6   Negative_Review                             515738 non-null  object
 7   Review_Total_Negative_Word_Counts           515738 non-null  int64
 8   Total_Number_of_Reviews                     515738 non-null  int64
 9   Positive_Review                             515738 non-null  object
 10  Review_Total_Positive_Word_Counts           515738 non-null  int64
 11  Total_Number_of_Reviews_Reviewer_Has_Given  515738 non-null  int64
 12  Reviewer_Score                              515738 non-null  float64
 13  Tags                                        515738 non-null  object
 14  days_since_review                           515738 non-null  object
 15  lat                                         512470 non-null  float64
 16  lng                                         512470 non-null  float64
dtypes: datetime64[ns](1), float64(4), int64(5), object(7)
memory usage: 66.9+ MB
```

Column name	Description	Data Type
Hotel_Address	Address of the hotel	object
Additional_Number_of_Scoring	There are also some guests who just made a scoring on the service rather than a review. This number indicates how many valid scores without review in there.	int

Column name	Description	Data Type
Review_Date	Date when reviewer posted the corresponding review.	datetime
Average_Score	Average Score of the hotel, calculated based on the latest comment in the last year.	float
Hotel_Name	Name of Hotel	object
Reviewer_Nationality	Nationality of Reviewer	object
Negative_Review	Negative Review the reviewer gave to the hotel.	object
Review_Total_Negative_Word_Counts	Total number of words in the negative review.	int
Total_Number_of_Reviews	Total number of valid reviews the hotel has.	int
Positive_Review	Positive Review the reviewer gave to the hotel.	object
Review_Total_Positive_Word_Counts	Total number of words in the positive review.	int
Total_Number_of_Reviews_Reviewer_Has_Given	Number of Reviews the reviewers has given in the past.	int
Reviewer_Score	Score the reviewer has given to the hotel, based on his/her experience	float
Tags	Tags reviewer gave the hotel.	object
days_since_review	Duration between the review date and scrape date.	object
lat	Latitude of the hotel	float
lng	Longtitude of the hotel	float

In [12]:
```
df.describe(include='all', datetime_is_numeric=True)
```

Out[12]:

	Hotel_Address	Additional_Number_of_Scoring	Review_Date	Average_Score	Hotel_Na
count	515738	515738.00000	515738	515738.00000	515
unique	1493	nan	NaN	nan	1
top	163 Marsh Wall Docklands Tower Hamlets London ...	nan	NaN	nan	Britar Internatic H Car W
freq	4789	nan	NaN	nan	4
mean	NaN	498.08184	2016-08-13 13:23:37.096383232	8.39749	N
min	NaN	1.00000	2015-08-04 00:00:00	5.20000	N
25%	NaN	169.00000	2016-02-23 00:00:00	8.10000	N
50%	NaN	341.00000	2016-08-15 00:00:00	8.40000	N
75%	NaN	660.00000	2017-02-09 00:00:00	8.80000	N
max	NaN	2682.00000	2017-08-03 00:00:00	9.80000	N
std	NaN	500.53847	NaN	0.54805	N

filter data

We filter the data or the following way:

1. use only reviews with a word count above 0
2. split the data into two dataframes (one for positive one for negative)
3. add a sentiment column
4. rename the text column (so it matches other dataframes)

In [13]:
```
# get reviews that have positive text
df_pos = df[df['Review_Total_Positive_Word_Counts'] > 0]
#get text data
df_pos = df_pos.filter(['Positive_Review'], axis=1)
#add sentiment
df_pos['sentiment'] = "POSITIVE"
#rename text colum to match amazon dataframe
df_pos = df_pos.rename(columns = {'Positive_Review': 'reviewText'}, inplace
display(df_pos)


df_neg = df[df['Review_Total_Negative_Word_Counts'] > 0]
#get text data
df_neg = df_neg.filter(['Negative_Review'], axis=1)
#add sentiment
df_neg['sentiment'] = "NEGATIVE"
#rename text colum to match amazon dataframe
df_neg = df_neg.rename(columns = {'Negative_Review': 'reviewText'}, inplace
display(df_neg)
```

	reviewText	sentiment
0	Only the park outside of the hotel was beauti...	POSITIVE
1	No real complaints the hotel was great great ...	POSITIVE
2	Location was good and staff were ok It is cut...	POSITIVE
3	Great location in nice surroundings the bar a...	POSITIVE
4	Amazing location and building Romantic setting	POSITIVE
...	...	...
515732	helpful staff allowed me to check in early as...	POSITIVE
515733	location	POSITIVE
515734	Breakfast was ok and we got earlier check in	POSITIVE
515736	The rooms are enormous and really comfortable...	POSITIVE
515737	staff was very kind	POSITIVE

476219 rows × 2 columns

	reviewText	sentiment
0	I am so angry that i made this post available...	NEGATIVE
2	Rooms are nice but for elderly a bit difficul...	NEGATIVE
3	My room was dirty and I was afraid to walk ba...	NEGATIVE
4	You When I booked with your company on line y...	NEGATIVE
5	Backyard of the hotel is total mess shouldn t...	NEGATIVE
...	...	...
515731	No parking Public parking garage is 15 Euro p...	NEGATIVE

	reviewText	sentiment
515733	no trolly or staff to help you take the lugga...	NEGATIVE
515734	The hotel looks like 3 but surely not 4	NEGATIVE
515735	The ac was useless It was a hot week in vienn...	NEGATIVE
515737	I was in 3rd floor It didn t work Free Wife	NEGATIVE

354697 rows × 2 columns

Now we need to merge the two separate data frames back together again

In [14]:
```
#combine
df_combined = pd.concat([df_pos, df_neg])
#shuffle
df_combined = df_combined.sample(frac=1, random_state=42)
#drop index
df_combined = df_combined.reset_index(drop=True)
df_combined
```

Out[14]:

	reviewText	sentiment
0	Extremely comfortable room with lots of great...	POSITIVE
1	Only one thing was on Wednesday 1st March it ...	NEGATIVE
2	Location very near to central train station a...	POSITIVE
3	at peak time breakfast the staff could not ke...	NEGATIVE
4	I liked almost everything from check in to ch...	POSITIVE
...	...	...
830911	Everything was nice	POSITIVE
830912	Just beautiful	POSITIVE
830913	Good staff warm reception	POSITIVE
830914	It was a long walk into Barcelona The view fr...	NEGATIVE
830915	The bed was comfy the room nice and big nice ...	POSITIVE

830916 rows × 2 columns

Conclusion

I now have a better understanding of the data in the dataset. I have filtered for what is useful and added a sentiment column. I feel confident that I can use this data in the next step.

Save DF

We are done with filtering and exploring the data of this dataset. Now we save it so it can be used in another file.

In [15]:
```
#save dataframe as csv to use it in another file
df_combined.to_csv(r'data/filtered/hotel_filtered.csv', index = False)
```

EDA Movie

Project

Several sites allow you to post a comment or opinion about a product, place, or event. Think of Facebook, Instagram, Youtube, or Reddit. However, on not all of these sites, you can see at a glance how many of these comments are written with a positive or negative mindset.

For many people or companies, it is useful to know what percentage of the reactions are positive or negative. You can't always tell this by looking at a Like system. The main question you should be able to answer with the help of this project is; Is a piece of text positive or negative?

Movie Dataset

This is a dataset with a lot of reviews about movies. It comes from kaggle.

source: [https://www.kaggle.com/nltkdata/movie-review (https://www.kaggle.com/nltkdata/movie-review)](https://www.kaggle.com/nltkdata/movie-review)

Document goal

In this paper, I want to gain more insight into the Movie dataset. And estimate if it is useful for achieving my project goal.

First We will start with importing libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

#set decimal point
pd.options.display.float_format = "{:.5f}".format
```

Let's start by loading the data set

In [2]:
```
df = pd.read_csv('data/raw/movie_review.csv')
```

Now that we have a data frame let's start looking at the data

In [3]:
```
1 df
```

Out[3]:

	fold_id	cv_tag	html_id	sent_id	text	tag
0	0	cv000	29590	0	films adapted from comic books have had plenty...	pos
1	0	cv000	29590	1	for starters , it was created by alan moore (...	pos
2	0	cv000	29590	2	to say moore and campbell thoroughly researche...	pos
3	0	cv000	29590	3	the book (or " graphic novel , " if you will ...	pos
4	0	cv000	29590	4	in other words , don't dismiss this film becau...	pos
...	...	...	...	...	...	...
64715	9	cv999	14636	20	that lack of inspiration can be traced back to...	neg
64716	9	cv999	14636	21	like too many of the skits on the current inca...	neg
64717	9	cv999	14636	22	after watching one of the " roxbury " skits on...	neg
64718	9	cv999	14636	23	bump unsuspecting women , and . . . that's all .	neg
64719	9	cv999	14636	24	after watching _a_night_at_the_roxbury_ , you'...	neg

64720 rows × 6 columns

Data ledger

Where is our data comming from?

Concerning fact	Movie review Data
Source	https://www.kaggle.com/nltkdata/movie-review (https://www.kaggle.com/nltkdata/movie-review)
Origin of the data	Kaggele, created by Bo Pang and Lillian Lee
rate of refreshing	NaN
date of the data	June, 2004
Period the data concerns	?
Size of data	**Size**: Approx. 8.72 MB, **Rows**: 64720, **Columns**: 6

The datasource does not have a legend telling what each column means. So we have to fill this in ourselves. But before we do that let's see if we are missing any data.

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

Out[4]:
```
fold_id    0
cv_tag     0
html_id    0
sent_id    0
text       0
tag        0
dtype: int64
```

It's nice that we don't miss any data! To guess what each column means we need to look a little more into the data.

In [5]:
```
1 df.describe(include="all")
```

Out[5]:

	fold_id	cv_tag	html_id	sent_id	text	tag
count	64720.00000	64720	64720.00000	64720.00000	64720	64720
unique	nan	1000	nan	nan	63652	2
top	nan	cv256	nan	nan	.	pos
freq	nan	166	nan	nan	123	32937
mean	4.54938	NaN	16074.09737	18.98118	NaN	NaN
std	2.85318	NaN	7175.28252	15.08369	NaN	NaN
min	0.00000	NaN	42.00000	0.00000	NaN	NaN
25%	2.00000	NaN	10613.00000	8.00000	NaN	NaN
50%	5.00000	NaN	15091.00000	16.00000	NaN	NaN
75%	7.00000	NaN	21865.00000	27.00000	NaN	NaN
max	9.00000	NaN	29867.00000	111.00000	NaN	NaN

Text and tag

This doesn't say a whole lot yet. However, we do already know that `text` is the review text. And that `tag` has only 2 unique values: `pos` and `neg`. This probably stands for **positive** and **negative** review. This is what we are interested in! But let's also look at the other columns

Fold Id

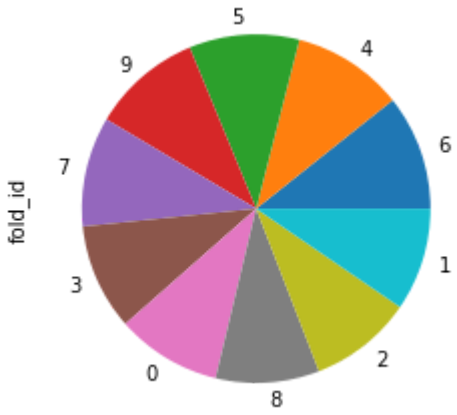
`Fold_id` will likely to be a subgrouping. To confirm this we look at how many unique values there are.

In [6]:
```
1 df.fold_id.unique()
```

Out[6]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=int64)
```

Now let's see how the distribution is

In [7]:
```
fig, ax = plt.subplots()
df['fold_id'].value_counts().plot(ax=ax, kind='pie')
plt.show()
```



I am confident in assuming that this is a subgrouping id.

CV tag

I can't imagine what this means. Probably it's also a kind of grouping let's see what kind of different values there are.

In [8]:
```
1 df.cv_tag.unique()
```

Out[8]:
```
array(['cv000', 'cv001', 'cv002', 'cv003', 'cv004', 'cv005', 'cv006',
       'cv007', 'cv008', 'cv009', 'cv010', 'cv011', 'cv012', 'cv013',
       'cv014', 'cv015', 'cv016', 'cv017', 'cv018', 'cv019', 'cv020',
       'cv021', 'cv022', 'cv023', 'cv024', 'cv025', 'cv026', 'cv027',
       'cv028', 'cv029', 'cv030', 'cv031', 'cv032', 'cv033', 'cv034',
       'cv035', 'cv036', 'cv037', 'cv038', 'cv039', 'cv040', 'cv041',
       'cv042', 'cv043', 'cv044', 'cv045', 'cv046', 'cv047', 'cv048',
       'cv049', 'cv050', 'cv051', 'cv052', 'cv053', 'cv054', 'cv055',
       'cv056', 'cv057', 'cv058', 'cv059', 'cv060', 'cv061', 'cv062',
       'cv063', 'cv064', 'cv065', 'cv066', 'cv067', 'cv068', 'cv069',
       'cv070', 'cv071', 'cv072', 'cv073', 'cv074', 'cv075', 'cv076',
       'cv077', 'cv078', 'cv079', 'cv080', 'cv081', 'cv082', 'cv083',
       'cv084', 'cv085', 'cv086', 'cv087', 'cv088', 'cv089', 'cv090',
       'cv091', 'cv092', 'cv093', 'cv094', 'cv095', 'cv096', 'cv097',
       'cv098', 'cv099', 'cv100', 'cv101', 'cv102', 'cv103', 'cv104',
       'cv105', 'cv106', 'cv107', 'cv108', 'cv109', 'cv110', 'cv111',
       'cv112', 'cv113', 'cv114', 'cv115', 'cv116', 'cv117', 'cv118',
       'cv119', 'cv120', 'cv121', 'cv122', 'cv123', 'cv124', 'cv125',
       'cv126', 'cv127', 'cv128', 'cv129', 'cv130', 'cv131', 'cv132',
       'cv133', 'cv134', 'cv135', 'cv136', 'cv137', 'cv138', 'cv139',
       'cv140', 'cv141', 'cv142', 'cv143', 'cv144', 'cv145', 'cv146',
       'cv147', 'cv148', 'cv149', 'cv150', 'cv151', 'cv152', 'cv153',
       'cv154', 'cv155', 'cv156', 'cv157', 'cv158', 'cv159', 'cv160',
       'cv161', 'cv162', 'cv163', 'cv164', 'cv165', 'cv166', 'cv167',
       'cv168', 'cv169', 'cv170', 'cv171', 'cv172', 'cv173', 'cv174',
       'cv175', 'cv176', 'cv177', 'cv178', 'cv179', 'cv180', 'cv181',
       'cv182', 'cv183', 'cv184', 'cv185', 'cv186', 'cv187', 'cv188',
       'cv189', 'cv190', 'cv191', 'cv192', 'cv193', 'cv194', 'cv195',
       'cv196', 'cv197', 'cv198', 'cv199', 'cv200', 'cv201', 'cv202',
       'cv203', 'cv204', 'cv205', 'cv206', 'cv207', 'cv208', 'cv209',
       'cv210', 'cv211', 'cv212', 'cv213', 'cv214', 'cv215', 'cv216',
       'cv217', 'cv218', 'cv219', 'cv220', 'cv221', 'cv222', 'cv223',
       'cv224', 'cv225', 'cv226', 'cv227', 'cv228', 'cv229', 'cv230',
       'cv231', 'cv232', 'cv233', 'cv234', 'cv235', 'cv236', 'cv237',
       'cv238', 'cv239', 'cv240', 'cv241', 'cv242', 'cv243', 'cv244',
       'cv245', 'cv246', 'cv247', 'cv248', 'cv249', 'cv250', 'cv251',
       'cv252', 'cv253', 'cv254', 'cv255', 'cv256', 'cv257', 'cv258',
       'cv259', 'cv260', 'cv261', 'cv262', 'cv263', 'cv264', 'cv265',
       'cv266', 'cv267', 'cv268', 'cv269', 'cv270', 'cv271', 'cv272',
       'cv273', 'cv274', 'cv275', 'cv276', 'cv277', 'cv278', 'cv279',
       'cv280', 'cv281', 'cv282', 'cv283', 'cv284', 'cv285', 'cv286',
       'cv287', 'cv288', 'cv289', 'cv290', 'cv291', 'cv292', 'cv293',
       'cv294', 'cv295', 'cv296', 'cv297', 'cv298', 'cv299', 'cv300',
       'cv301', 'cv302', 'cv303', 'cv304', 'cv305', 'cv306', 'cv307',
       'cv308', 'cv309', 'cv310', 'cv311', 'cv312', 'cv313', 'cv314',
       'cv315', 'cv316', 'cv317', 'cv318', 'cv319', 'cv320', 'cv321',
       'cv322', 'cv323', 'cv324', 'cv325', 'cv326', 'cv327', 'cv328',
       'cv329', 'cv330', 'cv331', 'cv332', 'cv333', 'cv334', 'cv335',
       'cv336', 'cv337', 'cv338', 'cv339', 'cv340', 'cv341', 'cv342',
       'cv343', 'cv344', 'cv345', 'cv346', 'cv347', 'cv348', 'cv349',
       'cv350', 'cv351', 'cv352', 'cv353', 'cv354', 'cv355', 'cv356',
       'cv357', 'cv358', 'cv359', 'cv360', 'cv361', 'cv362', 'cv363',
       'cv364', 'cv365', 'cv366', 'cv367', 'cv368', 'cv369', 'cv370',
       'cv371', 'cv372', 'cv373', 'cv374', 'cv375', 'cv376', 'cv377',
       'cv378', 'cv379', 'cv380', 'cv381', 'cv382', 'cv383', 'cv384',
```

```
'cv385', 'cv386', 'cv387', 'cv388', 'cv389', 'cv390', 'cv391',
'cv392', 'cv393', 'cv394', 'cv395', 'cv396', 'cv397', 'cv398',
'cv399', 'cv400', 'cv401', 'cv402', 'cv403', 'cv404', 'cv405',
'cv406', 'cv407', 'cv408', 'cv409', 'cv410', 'cv411', 'cv412',
'cv413', 'cv414', 'cv415', 'cv416', 'cv417', 'cv418', 'cv419',
'cv420', 'cv421', 'cv422', 'cv423', 'cv424', 'cv425', 'cv426',
'cv427', 'cv428', 'cv429', 'cv430', 'cv431', 'cv432', 'cv433',
'cv434', 'cv435', 'cv436', 'cv437', 'cv438', 'cv439', 'cv440',
'cv441', 'cv442', 'cv443', 'cv444', 'cv445', 'cv446', 'cv447',
'cv448', 'cv449', 'cv450', 'cv451', 'cv452', 'cv453', 'cv454',
'cv455', 'cv456', 'cv457', 'cv458', 'cv459', 'cv460', 'cv461',
'cv462', 'cv463', 'cv464', 'cv465', 'cv466', 'cv467', 'cv468',
'cv469', 'cv470', 'cv471', 'cv472', 'cv473', 'cv474', 'cv475',
'cv476', 'cv477', 'cv478', 'cv479', 'cv480', 'cv481', 'cv482',
'cv483', 'cv484', 'cv485', 'cv486', 'cv487', 'cv488', 'cv489',
'cv490', 'cv491', 'cv492', 'cv493', 'cv494', 'cv495', 'cv496',
'cv497', 'cv498', 'cv499', 'cv500', 'cv501', 'cv502', 'cv503',
'cv504', 'cv505', 'cv506', 'cv507', 'cv508', 'cv509', 'cv510',
'cv511', 'cv512', 'cv513', 'cv514', 'cv515', 'cv516', 'cv517',
'cv518', 'cv519', 'cv520', 'cv521', 'cv522', 'cv523', 'cv524',
'cv525', 'cv526', 'cv527', 'cv528', 'cv529', 'cv530', 'cv531',
'cv532', 'cv533', 'cv534', 'cv535', 'cv536', 'cv537', 'cv538',
'cv539', 'cv540', 'cv541', 'cv542', 'cv543', 'cv544', 'cv545',
'cv546', 'cv547', 'cv548', 'cv549', 'cv550', 'cv551', 'cv552',
'cv553', 'cv554', 'cv555', 'cv556', 'cv557', 'cv558', 'cv559',
'cv560', 'cv561', 'cv562', 'cv563', 'cv564', 'cv565', 'cv566',
'cv567', 'cv568', 'cv569', 'cv570', 'cv571', 'cv572', 'cv573',
'cv574', 'cv575', 'cv576', 'cv577', 'cv578', 'cv579', 'cv580',
'cv581', 'cv582', 'cv583', 'cv584', 'cv585', 'cv586', 'cv587',
'cv588', 'cv589', 'cv590', 'cv591', 'cv592', 'cv593', 'cv594',
'cv595', 'cv596', 'cv597', 'cv598', 'cv599', 'cv600', 'cv601',
'cv602', 'cv603', 'cv604', 'cv605', 'cv606', 'cv607', 'cv608',
'cv609', 'cv610', 'cv611', 'cv612', 'cv613', 'cv614', 'cv615',
'cv616', 'cv617', 'cv618', 'cv619', 'cv620', 'cv621', 'cv622',
'cv623', 'cv624', 'cv625', 'cv626', 'cv627', 'cv628', 'cv629',
'cv630', 'cv631', 'cv632', 'cv633', 'cv634', 'cv635', 'cv636',
'cv637', 'cv638', 'cv639', 'cv640', 'cv641', 'cv642', 'cv643',
'cv644', 'cv645', 'cv646', 'cv647', 'cv648', 'cv649', 'cv650',
'cv651', 'cv652', 'cv653', 'cv654', 'cv655', 'cv656', 'cv657',
'cv658', 'cv659', 'cv660', 'cv661', 'cv662', 'cv663', 'cv664',
'cv665', 'cv666', 'cv667', 'cv668', 'cv669', 'cv670', 'cv671',
'cv672', 'cv673', 'cv674', 'cv675', 'cv676', 'cv677', 'cv678',
'cv679', 'cv680', 'cv681', 'cv682', 'cv683', 'cv684', 'cv685',
'cv686', 'cv687', 'cv688', 'cv689', 'cv690', 'cv691', 'cv692',
'cv693', 'cv694', 'cv695', 'cv696', 'cv697', 'cv698', 'cv699',
'cv700', 'cv701', 'cv702', 'cv703', 'cv704', 'cv705', 'cv706',
'cv707', 'cv708', 'cv709', 'cv710', 'cv711', 'cv712', 'cv713',
'cv714', 'cv715', 'cv716', 'cv717', 'cv718', 'cv719', 'cv720',
'cv721', 'cv722', 'cv723', 'cv724', 'cv725', 'cv726', 'cv727',
'cv728', 'cv729', 'cv730', 'cv731', 'cv732', 'cv733', 'cv734',
'cv735', 'cv736', 'cv737', 'cv738', 'cv739', 'cv740', 'cv741',
'cv742', 'cv743', 'cv744', 'cv745', 'cv746', 'cv747', 'cv748',
'cv749', 'cv750', 'cv751', 'cv752', 'cv753', 'cv754', 'cv755',
'cv756', 'cv757', 'cv758', 'cv759', 'cv760', 'cv761', 'cv762',
'cv763', 'cv764', 'cv765', 'cv766', 'cv767', 'cv768', 'cv769',
'cv770', 'cv771', 'cv772', 'cv773', 'cv774', 'cv775', 'cv776',
'cv777', 'cv778', 'cv779', 'cv780', 'cv781', 'cv782', 'cv783',
```

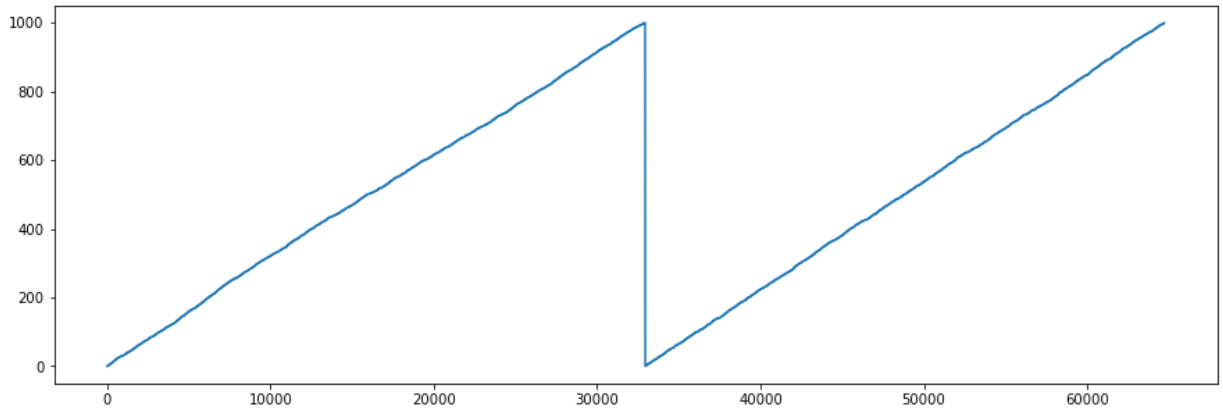
```
'cv784', 'cv785', 'cv786', 'cv787', 'cv788', 'cv789', 'cv790',
'cv791', 'cv792', 'cv793', 'cv794', 'cv795', 'cv796', 'cv797',
'cv798', 'cv799', 'cv800', 'cv801', 'cv802', 'cv803', 'cv804',
'cv805', 'cv806', 'cv807', 'cv808', 'cv809', 'cv810', 'cv811',
'cv812', 'cv813', 'cv814', 'cv815', 'cv816', 'cv817', 'cv818',
'cv819', 'cv820', 'cv821', 'cv822', 'cv823', 'cv824', 'cv825',
'cv826', 'cv827', 'cv828', 'cv829', 'cv830', 'cv831', 'cv832',
'cv833', 'cv834', 'cv835', 'cv836', 'cv837', 'cv838', 'cv839',
'cv840', 'cv841', 'cv842', 'cv843', 'cv844', 'cv845', 'cv846',
'cv847', 'cv848', 'cv849', 'cv850', 'cv851', 'cv852', 'cv853',
'cv854', 'cv855', 'cv856', 'cv857', 'cv858', 'cv859', 'cv860',
'cv861', 'cv862', 'cv863', 'cv864', 'cv865', 'cv866', 'cv867',
'cv868', 'cv869', 'cv870', 'cv871', 'cv872', 'cv873', 'cv874',
'cv875', 'cv876', 'cv877', 'cv878', 'cv879', 'cv880', 'cv881',
'cv882', 'cv883', 'cv884', 'cv885', 'cv886', 'cv887', 'cv888',
'cv889', 'cv890', 'cv891', 'cv892', 'cv893', 'cv894', 'cv895',
'cv896', 'cv897', 'cv898', 'cv899', 'cv900', 'cv901', 'cv902',
'cv903', 'cv904', 'cv905', 'cv906', 'cv907', 'cv908', 'cv909',
'cv910', 'cv911', 'cv912', 'cv913', 'cv914', 'cv915', 'cv916',
'cv917', 'cv918', 'cv919', 'cv920', 'cv921', 'cv922', 'cv923',
'cv924', 'cv925', 'cv926', 'cv927', 'cv928', 'cv929', 'cv930',
'cv931', 'cv932', 'cv933', 'cv934', 'cv935', 'cv936', 'cv937',
'cv938', 'cv939', 'cv940', 'cv941', 'cv942', 'cv943', 'cv944',
'cv945', 'cv946', 'cv947', 'cv948', 'cv949', 'cv950', 'cv951',
'cv952', 'cv953', 'cv954', 'cv955', 'cv956', 'cv957', 'cv958',
'cv959', 'cv960', 'cv961', 'cv962', 'cv963', 'cv964', 'cv965',
'cv966', 'cv967', 'cv968', 'cv969', 'cv970', 'cv971', 'cv972',
'cv973', 'cv974', 'cv975', 'cv976', 'cv977', 'cv978', 'cv979',
'cv980', 'cv981', 'cv982', 'cv983', 'cv984', 'cv985', 'cv986',
'cv987', 'cv988', 'cv989', 'cv990', 'cv991', 'cv992', 'cv993',
'cv994', 'cv995', 'cv996', 'cv997', 'cv998', 'cv999'], dtype=object)
```

It's just a number from 0 to 999 with `cv` in front of it. Let's replace this with just the number. That way we can plot it easier.

In [9]:
```
#clean out 'cv'
df['cv_tag'] = df['cv_tag'].str.replace("cv", "")
#change to numeric type
df['cv_tag'] = pd.to_numeric(df['cv_tag'])
```

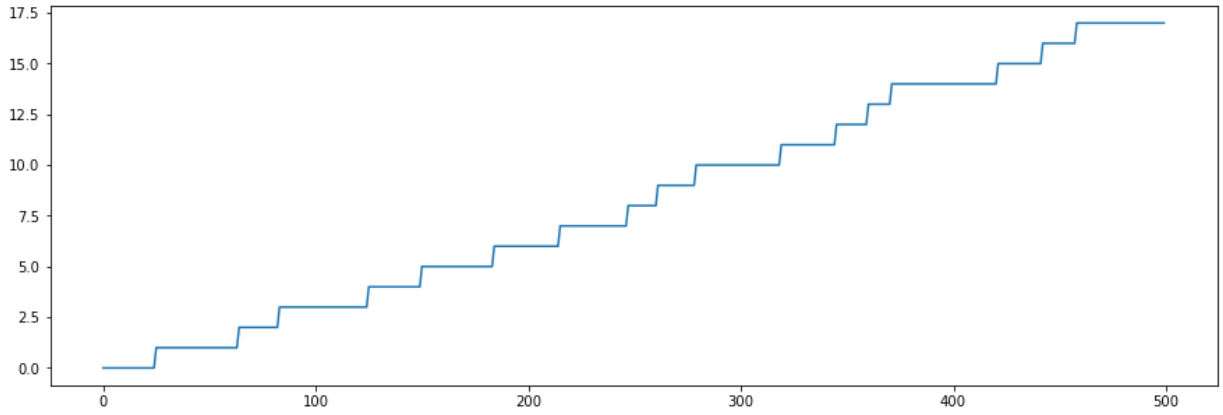
I still feel at this point that this is a grouping. Let's see how it looks in a graph.

In [10]:
```
fig, axs = plt.subplots()

fig.set_figheight(5)
fig.set_figwidth(15)

df['cv_tag'].plot(ax=axs, kind='line')
plt.show()
```



It continues to add up over the length of the data. This happens twice. And there are also 2 types of tags positive and negative. This is probably related to each other. Perhaps a zoomed-in small piece will give more insight.

In [11]:
```
fig, axs = plt.subplots()

fig.set_figheight(5)
fig.set_figwidth(15)

df['cv_tag'][:500].plot(ax=axs, kind='line')
plt.show()
```



The values seem to be divided into blocks. That's all I can see about it at the moment.

HTML id

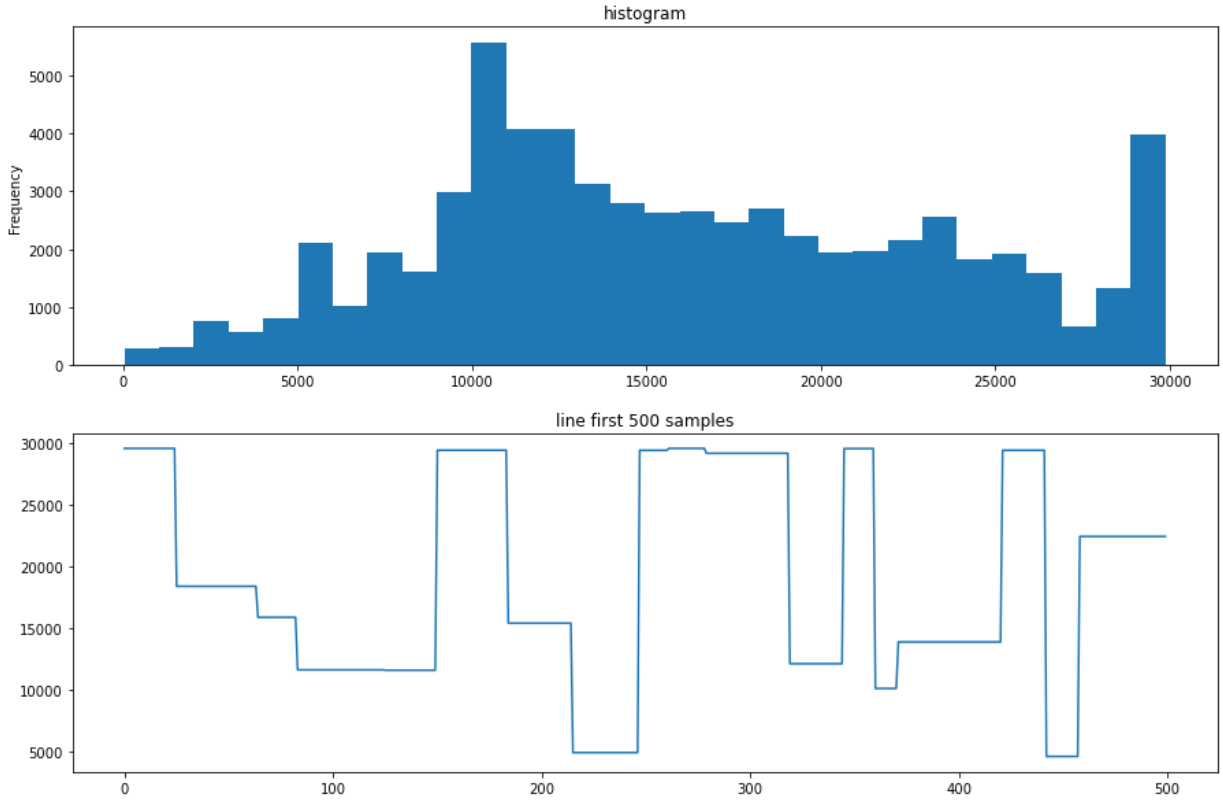
Html is a language in which websites are created. But whether this is related to this dataset? Let's start by looking at what different values there are.

In [12]:
```
print(sorted(df.html_id.unique()))
```

```
[42, 49, 50, 57, 74, 150, 984, 1198, 1250, 1381, 1576, 1730, 1754, 1841, 194
1, 1989, 2029, 2053, 2068, 2107, 2145, 2230, 2253, 2269, 2287, 2389, 2396, 24
50, 2556, 2618, 2666, 2673, 2693, 2695, 2856, 2877, 2895, 2915, 2953, 3006, 3
087, 3092, 3108, 3193, 3257, 3343, 3367, 3385, 3416, 3421, 3515, 3541, 3675,
3793, 3832, 3903, 3954, 3967, 3968, 3977, 4041, 4101, 4111, 4122, 4136, 4230,
4264, 4295, 4348, 4365, 4367, 4389, 4471, 4526, 4659, 4719, 4776, 4782, 4876,
4938, 4967, 4968, 4978, 4992, 5006, 5007, 5009, 5010, 5012, 5016, 5043, 5045,
5046, 5054, 5063, 5079, 5088, 5090, 5107, 5108, 5126, 5137, 5152, 5164, 5168,
5179, 5221, 5262, 5297, 5301, 5306, 5311, 5312, 5338, 5358, 5367, 5376, 5378,
5380, 5383, 5389, 5393, 5396, 5414, 5416, 5418, 5423, 5425, 5450, 5461, 5524,
5562, 5578, 5581, 5583, 5618, 5619, 5622, 5626, 5627, 5641, 5644, 5649, 5674,
5682, 5702, 5710, 5713, 5754, 5778, 5788, 5792, 5793, 5794, 5806, 5866, 5870,
5873, 5928, 5947, 5963, 5964, 5972, 6014, 6021, 6027, 6035, 6079, 6169, 6170,
6239, 6250, 6359, 6370, 6385, 6463, 6495, 6500, 6522, 6534, 6552, 6559, 6621,
6647, 6649, 6653, 6654, 6678, 6721, 6751, 6778, 6833, 6836, 6875, 6895, 6923,
6957, 6964, 6965, 6997, 7033, 7050, 7078, 7082, 7085, 7110, 7188, 7208, 7216,
7223, 7236, 7245, 7281, 7282, 7308, 7367, 7368, 7375, 7392, 7394, 7398, 7400,
7410, 7421, 7424, 7428, 7435, 7436, 7439, 7453, 7479, 7502, 7538, 7540, 7543,
7571, 7574, 7630, 7709, 7717, 7751, 7773, 7791, 7804, 7845, 7846, 7869, 7884,
```

There are very many different values. Maybe a histogram or a line diagram will give you more insight.

In [13]:
```
fig, axs = plt.subplots(2)

fig.set_figheight(10)
fig.set_figwidth(15)

df['html_id'].plot(ax=axs[0], kind='hist',bins=30, title="histogram")
df['html_id'][:500].plot(ax=axs[1], kind='line', title= "line first 500 samp
plt.show()
```



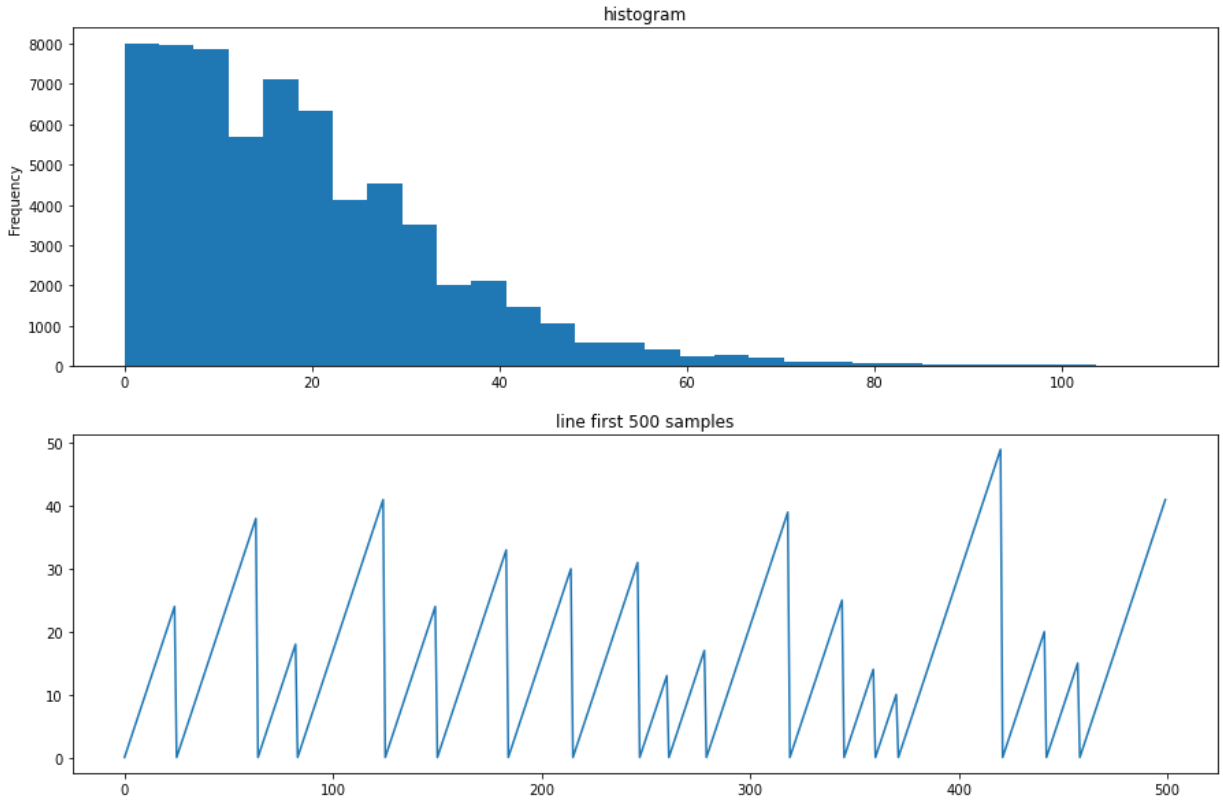
You can see that ids between 10,000 and 15,000, and just under 30,000 are more common than other ids. In the line graph, you can see that this data also appears to be divided into blocks, but so there is no smooth flow to be seen. Other than that, I can't get much information from this about what this column is about.

I suspect that this dataset was scrapped from the web and that this column refers to an html element or something like that. I cannot confirm this.

Sent id

Sending can refer to many things. But if we keep the scraping scenario in mind, it might have to do with sending the scraped data. Let's also put this data into a histogram and into a line diagram.

In [14]:
```
fig, axs = plt.subplots(2)

fig.set_figheight(10)
fig.set_figwidth(15)

df['sent_id'].plot(ax=axs[0], kind='hist',bins=30, title="histogram")
df['sent_id'][:500].plot(ax=axs[1], kind='line', title= "line first 500 samp
plt.show()
```



The lower the number the more common it is. That is one conclusion we can draw from the histogram. This can also be explained if we look at the line diagram. It is always counted from zero upwards. Again, we see that the data seems to be divided into blocks.

Column information

Note: This is an assumption, actual information is not available.

Column name	Description	Data Type
fold_id	part (fold) index of data	int
cv_tag	?	int

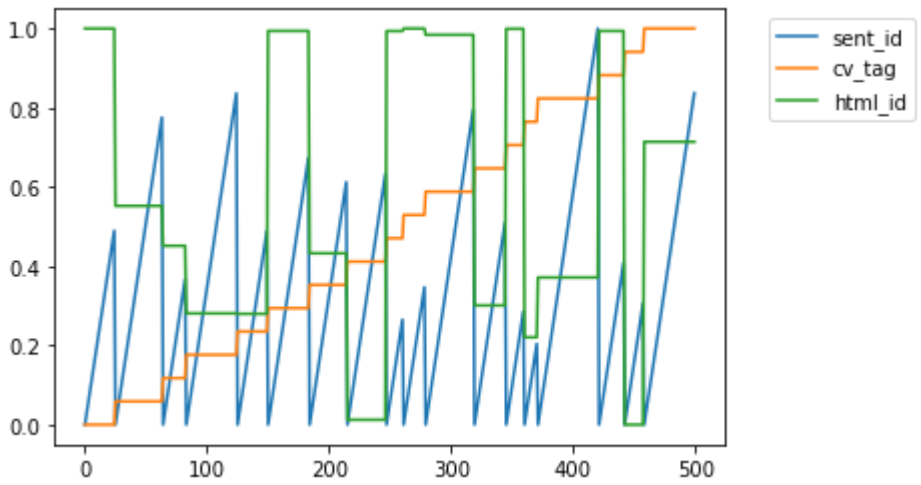
Column name	Description	Data Type
html_id	id of the html element data was scraped from	int
sent_id	id of the message sended with	int
text	The text of the review	object
tag	The sentiment tag (positive or negative)	object

Data blocks

To see if the data is distributed in the same blocks we are going to put the data in the same graph. For this we need to normalize the data. for this we create a function.

In [15]:
```
def normalise(series):
    mini = min(series)
    maxi = max(series)
    normalised = (series - mini) / (maxi - mini)
    return normalised;
```

In [16]:
```
fig, ax = plt.subplots()

sent_id = df['sent_id'][:500]
normalise(sent_id).plot(ax=ax, kind='line')

cv_tag = df['cv_tag'][:500]
normalise(cv_tag).plot(ax=ax, kind='line')

html = df['html_id'][:500]
normalise(html).plot(ax=ax, kind='line')

plt.legend(bbox_to_anchor=(1.05, 1),loc='upper left')
plt.show()
```



The data is indeed divided into the same blocks. What this means I do not know. I assume this has something to do with scrapping. This data does not seem interesting for the project so let's look further into `tag` and `text` .

Column information

Column name	Description	Data Type
overall	the overal score a product	float
vote	helpfull votes of review	float
verified	reviewer verified state	bool
reviewTime	date of the review	datetime
reviewerID	The ID of the reviewer	object
asin	The ID of the product	object
reviewerName	The name of the reviewer	object
summary	The summary of the review	object
unixReviewTime	date of the review in unix format	int
style	A summary of product specifications, e.g., "Color" or "Size"	object
image	Images the reviewer uploaded after recieving the product	object

Tag and Text

We are going to start by renaming the columns so that they are the same in each data frame

In [17]:
```
df = df.rename(columns = {'text': 'reviewText', 'tag': 'sentiment'}, inplace
df = df.filter(['reviewText' , 'sentiment'], axis=1)
df
```

Out[17]:

	reviewText	sentiment
0	films adapted from comic books have had plenty...	pos
1	for starters , it was created by alan moore (...	pos
2	to say moore and campbell thoroughly researche...	pos
3	the book (or " graphic novel , " if you will ...	pos
4	in other words , don't dismiss this film becau...	pos
...	...	...
64715	that lack of inspiration can be traced back to...	neg
64716	like too many of the skits on the current inca...	neg
64717	after watching one of the " roxbury " skits on...	neg
64718	bump unsuspecting women , and . . . that's all .	neg
64719	after watching _a_night_at_the_roxbury_ , you'...	neg

64720 rows × 2 columns

The column names are now correct. But we change the values of sentiment so that they are the same in each file.

In [18]:
```
df['sentiment'] = df['sentiment'].replace('pos', 'POSITIVE')
df['sentiment'] = df['sentiment'].replace('neg', 'NEGATIVE')
df
```

Out[18]:

	reviewText	sentiment
0	films adapted from comic books have had plenty...	POSITIVE
1	for starters , it was created by alan moore (...	POSITIVE
2	to say moore and campbell thoroughly researche...	POSITIVE
3	the book (or " graphic novel , " if you will ...	POSITIVE
4	in other words , don't dismiss this film becau...	POSITIVE
...	...	...
64715	that lack of inspiration can be traced back to...	NEGATIVE
64716	like too many of the skits on the current inca...	NEGATIVE
64717	after watching one of the " roxbury " skits on...	NEGATIVE
64718	bump unsuspecting women , and . . . that's all .	NEGATIVE
64719	after watching _a_night_at_the_roxbury_ , you'...	NEGATIVE

64720 rows × 2 columns

Conclusion

I now have a better understanding of the data in the dataset. I have filtered for what is useful and added a sentiment column. I feel confident that I can use this data in the next step.

Save DF

We are done with filtering and exploring the data of this dataset. Now we save it so it can be used in another file.

In [19]:
```
#save dataframe as csv to use it in another file
df.to_csv(r'data/filtered/movie_filtered.csv', index = False)
```

EDA combined

Project

Several sites allow you to post a comment or opinion about a product, place, or event. Think of Facebook, Instagram, Youtube, or Reddit. However, on not all of these sites, you can see at a glance how many of these comments are written with a positive or negative mindset.

For many people or companies, it is useful to know what percentage of the reactions are positive or negative. You can't always tell this by looking at a Like system. The main question you should be able to answer with the help of this project is; Is a piece of text positive or negative?

Datasets

This notebook uses several datasets. These have been filtered and modified in other notebooks. Look at the other notebooks for more information about the filtering process

Amazon Dataset

This is a dataset with all reviews coming from the site Amazon. source: https://nijianmo.github.io/amazon/index.html (https://nijianmo.github.io/amazon/index.html)

Movie Dataset

This is a dataset with a lot of reviews about movies. It comes from kaggle. source: https://www.kaggle.com/nltkdata/movie-review (https://www.kaggle.com/nltkdata/movie-review)

Hotel Dataset

This datasets contains data from a lot of hotels. Each reviewer left a positive and a negative review. The data was scraped from https://Booking.com (https://Booking.com). It comes from kaggle. source: https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe (https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe)

Document goal

In this paper, we want to better understand the data. We are going to see what relationships we see and what the word usage is like in certain subsets.

First We will start with importing libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

#set decimal point
pd.options.display.float_format = "{:.3f}".format
```

Loading the dataframes

In [2]:
```
df_amazon = pd.read_csv('data/filtered/amazon_filtered.csv')
df_hotel = pd.read_csv('data/filtered/hotel_filtered.csv')
df_movie = pd.read_csv('data/filtered/movie_filtered.csv')
```

Data manipulation

It is probably interesting to know which review comes from which source. To make this easier, we are adding a source column.

In [3]:
```
df_amazon['source'] = 'AMAZON'
df_hotel['source'] = 'HOTEL'
df_movie['source'] = 'MOVIE'
```

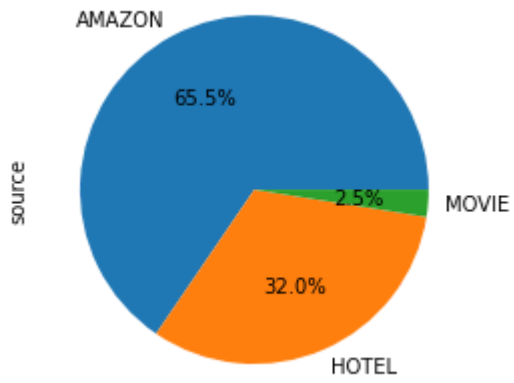
Concatenate dataframes

From now on, I want to consider the data as one dataset. So we are going to concatenate the different data frames.

In [4]:
```
#Concatenate
df_combined = pd.concat([df_amazon, df_hotel, df_movie])
#shuffle
df_combined = df_combined.sample(frac=1, random_state=42)
#drop index
df_combined = df_combined.reset_index(drop=True)
```

Let's look at the distribution of source.

In [5]:
```
fig, ax = plt.subplots()
df_combined['source'].value_counts().plot(ax=ax, kind='pie', autopct='%1.1f%
plt.show()
```



We see that the distribution is not quite fair. But that's not a problem for now.

wordcount

Also let's add a word count column. Because I am interested in how many words a review consists of.

In [6]:
```
df_combined['wordCount'] = df_combined['reviewText'].str.split().str.len()
```

In [7]:
```
1 df_combined
```

Out[7]:

	reviewText	sentiment	source	wordCount
0	saw in stock at TRU Ncea Commando 30th Anniver...	POSITIVE	AMAZON	288.000
1	I really wanted to like this gun but it arrive...	NEGATIVE	AMAZON	24.000
2	Staff has no hospitality skills they are just...	NEGATIVE	HOTEL	19.000
3	the location	POSITIVE	HOTEL	2.000
4	Very noisy and expensive breakfast	NEGATIVE	HOTEL	5.000
...	...	...	...	...
2594085	This is such a cute toy and it's easy to use a...	POSITIVE	AMAZON	36.000
2594086	Adorable, cute and soft. My 4 year old had a ...	POSITIVE	AMAZON	31.000
2594087	staff were quite friendly and helpful anythin...	POSITIVE	HOTEL	14.000
2594088	We noticied a lot of small but disgusting thi...	NEGATIVE	HOTEL	42.000
2594089	Stayed only one night pleasant experience	POSITIVE	HOTEL	6.000

2594090 rows × 4 columns

What is the mean word count distributed by sentiment?

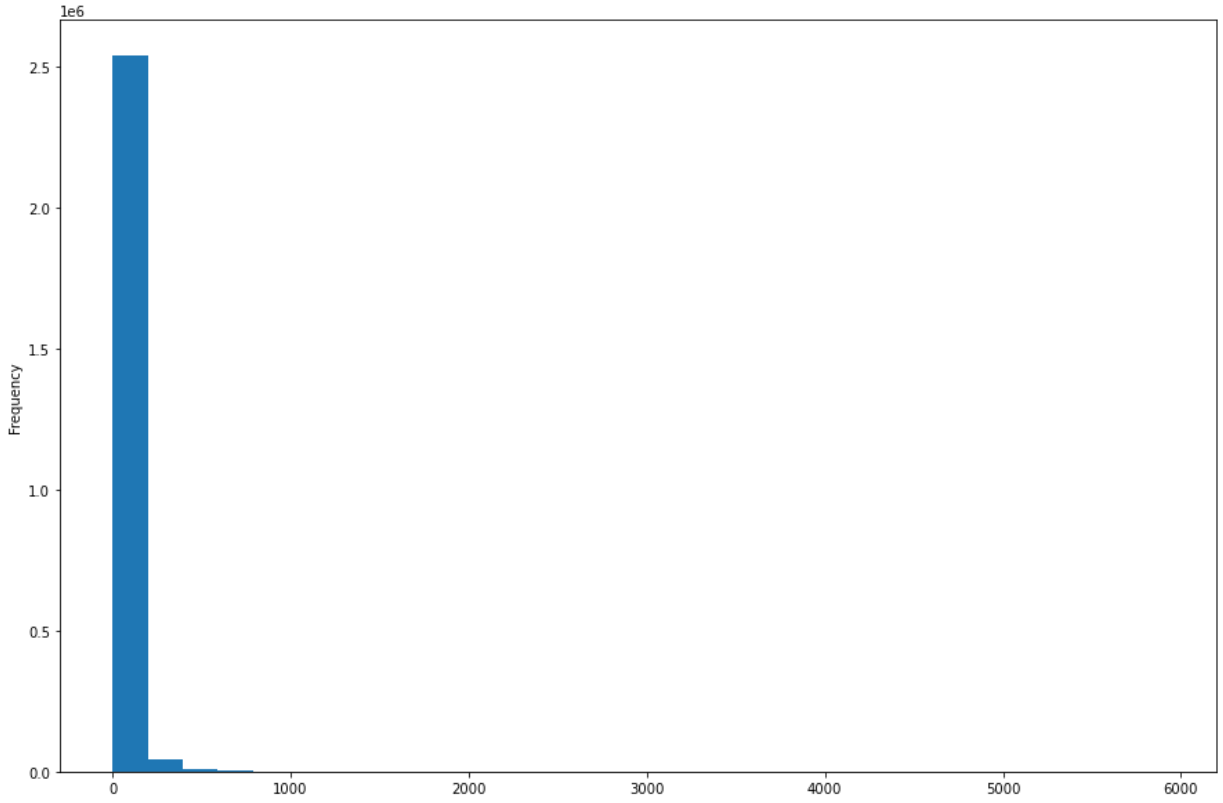
In [8]:
```
1 print('Positive word count:', df_combined[df_combined['sentiment']=='POSITIV
2 print('Negative word count:', df_combined[df_combined['sentiment']=='NEGATIV
```

```
Positive word count: 34.69408720260936
Negative word count: 32.5224985879959
```

It is noticeable that positive text consists of more words on average.

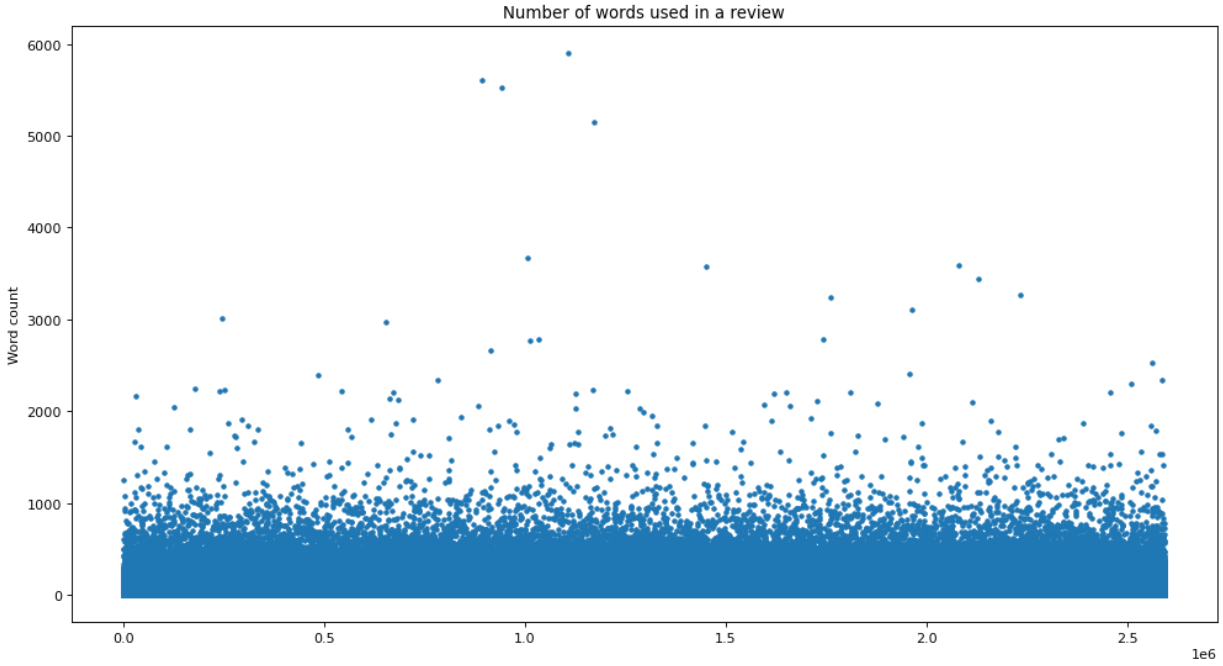
Let us now visualize the word count distribution. Using a histogram.

In [9]:
```
fig, ax = plt.subplots()

fig.set_figheight(10)
fig.set_figwidth(15)

df_combined['wordCount'].plot(ax=ax, kind='hist', bins=30)
plt.show()
```



It seems that most reviews have under 500 words. But the histogram goes up to 6000 words! Let's do a scater plot to identify outliers.

In [10]:
```
#make figure bigger
figure(figsize=(15, 8), dpi=80)

#add text
plt.ylabel('Word count')
plt.title("Number of words used in a review")

plt.plot(df_combined['wordCount'], '.');

plt.show()
```



I can not imagine a good training review that has over `750` words. How does a review that has over `750` words looks like?

In [11]:
```
#Get high word count
high_wordcount = df_combined[df_combined['wordCount'] > 750]

print('Reviews that have over 750 words:', len(high_wordcount))
```

```
Reviews that have over 750 words: 1522
```

In [12]:
```
for ind, review in enumerate(high_wordcount.reviewText[:3]):
    print(ind, ':',review)
    print( '\n'*10)
```

```
0 : For background before I begin my review, I am an electrical engineer who en
joys teaching his children about electronics.  I love educational toys and scie
nce toys, but I do tend to be critical of toys that I feel dumb down the scienc
e too much.  I had a friend who bemoaned that microprocessor development kits l
ike Arduino teach thousands of kids to blink an LED with a million transistors,
and yet they are left not knowing how to blink an LED with one transistor.  Tha
t being said, kits like Arduino has introduced many to the "maker" movement and
engineering in general, so I have no problem with the concept of simplified too
ls for beginners.

All of this introduction brings me to "Little Bits".  I gave this set to my 6 y
ear old and 8 year old to play with, starting with the 8 year old.  He loves hi
s <a data-hook="product-link-linked" class="a-link-normal" href="/Elenco-S
nap-Circuits-SC-750R-Student-Training-Program/dp/B000IXMP6Q/ref=cm_cr_arp_d_rvw
_txt?ie=UTF8">Elenco Snap Circuits SC-750R Student Training Program</a>, and ma
kes all sorts of cool things, so I thought he would be the best to evaluate it.

When you open the box, you are ready to go.  The high quality box has a magneti
c flap for closure, and inside is a full color glossy manual and all of the "li
ttle bits".  Each of the little bits has only an input and and output, so no mu
ltiple connections or wires to make.  There are blue (for power), pink (for con
trol), orange (wires), and green (for actions).

You get a custom printed 9V battery (no searching for batteries - thank you!),
and a bunch of modules.  There is 1 power module.  This module has a plug for t
he 9V battery, a switch, and a red indicator LED.  For you engineers, the modul
e has an ST LD1117 linear regulator on it.

There are 3 pink control modules, a button, a light sensor, and a dimmer.  I re
ally do wish they would use the technical term instead of the popular one on th
ese modules.  For instance, the button module is a simple on-off button.  Why n
ot call it a switch?  The dimmer module is a potentiometer/variable resistor.
The toy is so easy to use, why not let kids learn the right terms while they pl
ay?  The light sensor is a nice little device (though they could have called it
a photodetector or photodiode).  The light sensor module has a switch to let it
reach to light or dark, and a small potentiometer to adjust the sensitivity.  T
his is rather weakly executed because you need the tiny screwdriver (included)
to adjust the sensitivity.  My 6 year old had some trouble with dexterity adjus
ting it.  These are also easy to break (believe me), so if the toy is designed
to be simple, this could be a weak link.

There are wire modules which enable you to extend the circuit.  This is a nice
feature because you can actually wire the little bits into funny circuits like
hand buzzers, artwork like windmills, flashlights, robots, etc.  You get 2 wire
modules which can be added anywhere in the chain.

For action, you get a bright LED, a bar graph, a buzzer, and a DC motor.  I'm g
lad they didn't call the bright LED  a "shiny light" or the DC motor "spinning
thing" - they actually used the right term (unlike the sensors).  The bright LE
D is good for flashlights, and the bar graph is nice to use as a current meter
to see visually how much current is being supplied after the potentiometer to y
our creations.  The buzzer is quite annoying - your kids will love it.  The DC
motor is rather slow, but seems to work well for various projects.  There is a
```

"motor mate" which allows you to connect the motor to a LEGO axle or other thin gs. This enables you to build cool larger projects that move, a great idea.

Each of the bits magnetically click together so they cannot be put together inc orrectly (they repel if connected backwards). You start at Blue (power), pick control (pink), and anything after the control is affected by it (green). So i f you chain goes Power - LED - Switch, then the LED will always be on. But if it goes Power - Switch - LED, then the LED is controlled by the switch. You do n't need to terminate the end as the modules hook in parallel (except for thing s controlled by the pink modules). It really is foolproof.

There are a bunch of fun projects in the kit such as light controlled vehicles, doorbell, windmill, etc. Most require some arts and crafts beyond just the kit (some cardboard, tape, cutting, glue, etc).

I had decided to leave the ultimate verdict to my kids. My 8 year old tired o f it quickly. He said that his Snap Circuits teaches him more about electronic s, lets him change resistance and capacitance to see the change, and has more p ossibilities. My 6 year old has difficulty building the large Snap Circuits cr eations, so this was fun for him to get involved in the electronics of his olde r brother without the frustration. He has been playing with it for hours, grab bing cardboard to build a car so he car so he can attach it to the motor, etc. He emphatically gave it 5 stars.

However, I'm the parent and educator, so overall, I give it 4 stars. The entir e kid is a quality production and is fun for a while, but older kids may tire o f it quickly. The portable nature (unlike snap circuits) means you can build c ool portable toys like hand buzzers and robots, but the modular nature means yo u won't learn much about resistors, capacitors, diodes, etc. As my friend obse rved about Arduino, you may blink an LED with a million transistors, but can't do it with 1. I also have a tough time with the cost compared to Snap Circuits (which you should also consider). This is the "basic" kit, but still costs $10 0. I can honestly say that while my 6 year old likes it and it is well done, i t is not educational enough that I would have ever purchased this myself, and I don't intend to buy more of them.

Apart from addressing the high cost, I would suggest the designers find subtle ways to include lessons about what the kid is actually doing into the kit. For instance, silkscreen the schematic symbol of each major component on the board. Rename the modules with their technical terms. Put some explanation in the man ual as to how the circuits work and why they work. Kids shouldn't just build t hings, but understand what they built. This is the biggest problem with many o f these kid science and STEM activities in general - I see kids building all so rts of stuff not knowing. They build great projects but have no clue how they actually work. I would hope that if time was spent in an activity, the kid sho uld learn something from it.

Overall, I have to say that while I personally wouldn't buy this again, there a re plenty of people who will be quite happy with this kit as a fun toy. It is high quality and works quite well, so I am giving it a positive rating of 4 sta rs. It might be just the thing you are looking for (indeed, not everyone even wants the complexity of Snap Circuits), so read my review and if it sounds like it's for you, then I'm sure your kids will enjoy it.

1 : While shopping one day at a major retail store, I found this product on sale. After a phone call to my wife and some research by her, we found that many of the reviews across the internet were not very good. I was going to rent a slide similar to this from a party supply rental (the rental was bigger of course) for my child's birthday party. However, the rental fee would have been the same as this slide, so I decided to buy it, thinking if it lasted the season I would be ahead.

The slide was for my child who is about to turn 4. Internet research indicated that these slides tend to dry rot along the seams after a season, they tear easily, or they just plain leak. As a matter of fact, while standing in the customer service line at the store where I purchased this item, an older gentleman was returning the Banzai Double Drop Raceway. He told me that the slide tore at the seams the first time his grandkids used it (and that model has a 400 pound weight limit).

The initial set-up of the slide took me about 30 minutes. Set-up of the slide can be very easy or very difficult, depending mainly on the kid factor. If you are an adult by themselves, taking your time, the slide would be a breeze. If you have kids screaming, yelling, and trying to help, setting up the slide the first time is about as fun as playing patty-cake with an octopus. After the first time the slide is set-up, it only takes a few minutes. The first time you set it up, it takes a little time to learn where to properly position the fan, to take the paper off the decals, to figure out how to Velcro the water hoses to the slide, and how to position the slide if your lawn isn't level.

When I unpackaged the slide, it wasn't as cheaply built as I expected. The slide seems to be built out of a canvas like material that seems pretty durable. I did notice some air bubbles coming through at the seams, but there are stickers on the slide that say this is normal. There is a port that you can control how fast the air exits the slide with a string. A little adjustment of this port (to make it larger) and the air bubbles stopped.

The first time I set it up, I did not use any of the stabilizers or stakes that came with the slide. Even without the safety equipment, the slide seemed very stable. I was able to tip the slide over if I put my whole 230 pounds into it. A small child at the top did not seem to affect its stability in the slightest.

The climb to the top of the slide is a little more than 6 feet. The ladder is made up of "holes" that the child can stick their feet into as well as handles. My 4 year old was able to climb the "ladder" with minimal (if any) assistance.

So far, I am pleased with my purchase. The slide seems to be of good quality and is quite stable and of good design. The paper used to protect the decals during shipping is very difficult to remove. After the first time, the set-up is quite easy. I will update this review as I (or if I) encounter issues with the quality of the product.

Overall, this is definitely a product I would recommend for kids 3 and over.

Update #1: I used the slide today in light winds (under 10 MPH) without any of the restraints. It was quite a sight to see a 20 foot slide tumbling across th e yard in the wind. I used the stakes and water weights provided. The slide d idn't move anywhere, but it was still a little wobbly on top. I suggest connec ting the water bags to the slide first and then fill the bags with water.

Update#2: My child's birthday party came and went. The slide survived a thras hing by 4 young children who couldn't get enough of it. All the children were under 6 years of age. The 6 year old child is very tall for their age. I noti ced that for a tall 6 year old, the flat portion of the slide seemed too short and they would literally slide through the barrier at the end of the slide and onto the ground. They thought it was funny, but I could see that repeatedly hi tting the end barrier over and over might cause a tear in the slide. I have ye t to have anyone over 60 pounds on the slide, but I wonder if someone pushing t he 200 pound weight limit would have a problem stopping before the end barrier on the slide (then again, the kid could be just being a kid and hitting the end on purpose). Like I said, the slide survived just fine and it looks like it wi ll survive season #1.

Update #3: The slide survived season #2. At 230 pounds, I opted to go down th e slide despite the 200 pound weight limit. The slide didn't break, but it had troubles staying inflated under my weight. The slide is still holding up well despite repeated beatings from small children. The only thing worth mentioning is that more water seems to be leaching into the inside of the slide this year, requiring the slide to be dried out for a day or two before putting it away. O verall, I am still pleased with the purchase.

2 : Ring Side Report-Board Game Review of Diamonds

Originally posted at www.throatpunchgames.com, a new idea everyday!

Product Diamonds

Producer-Stronghold games

Price $ 25 here http://www.amazon.com/dp/B00NFSBU9Y/ref=sr_ph_1?m=A7YAR2WDYOPTK &ie=UTF8&qid=1440129851&sr=sr-1&keywords=diamonds (http://www.amazon.com/dp/B00 NFSBU9Y/ref=sr_ph_1?m=A7YAR2WDYOPTK&ie=UTF8&qid=1440129851&sr=sr-1&keywords=dia monds)

Set-up/Play/Clean-up 30-45 minutes (2-6players)

Type-Euro

Depth-Light

TL; DR-Best parts of several trick tacking games. 92.5%

basics

Basics-Time to throw Eucker, Hearts, and Spades into a blender! Diamonds is a trick taking game that combines the best of all of the above. Players are dealt 10 cards, and the dealer will decide to trade one to three cards. All players then choose that many cards from their hand and pass them to their neighbor. Next, the player to the left of the dealer will place one card down. These cards have values between one to 15 and have the four suits found on any normal deck of cards: hearts, spades, diamonds, and clubs. Each player then has to place a card of the leaders suit, if they have any, or play any other card, if they don't have the same suit as this tricks lead card. Here is where the game becomes more than just a trick taking game. Each suit has a power associated with it. Diamond cards place a diamond behind a small screen called your vault. Hearts place a diamond in front of the vault in your vault. Spades take a diamond from the front and place it behind your vault. Clubs steal a diamond from in front of another players vault and places it in front of yours. Whoever played the highest card with the leads suit gets to take all the played cards and set them next to his or her screen and then take that suits action. If you couldn't play a card with the leads suit, you just take the action associated with your suit. Playing off won't get you cards for the rounds end, but it does get you whatever power the card you played has. Whoever won that trick then becomes the next lead player for the next trick. After 10 tricks each round, all players separate their cards into four piles based on the suits. Whoever has the most number of each suit gets to take that suits power again. If you didn't get any tricks, and thus have no cards, you get two free diamonds placed right into your vault instead. Play then continues with a new dealer. After each player deals one or two times, depending on the player count, each player counts their diamonds with diamonds in your vault worth two points and those in front being worth one. The player with the most points wins!

2015-08-21_1440178740

Mechanics Im from Michigan, so I knew this game from another game called Eucker. Eucker is fun, but it lacks depth. This game is amazingly deep for a trick taking card game. Sometimes you do better by playing off than ever winning a single trick. Sometimes, you need to win every trick. That evolving strategy is amazing. Also, the game isn't hard to play. I do love me some 8-hour, math fueled, Euros where I build cars, but you will learn this game in under 3 minutes, master it in 10, and have a chance of winning in 15. Honestly, this is a well done game. 5/5

Theme-Theme is a hard concept in your average trick taking game. Whats here is ok. There really isn't a story here. But then again, Im not really looking for one. Id like more, as Ive seen some reskinned trick taking card games with more theme, but I didn't expect too much going in. The components are nice and do build a bit of a world, but don't play this game if you need something like Dark Moons story. 3.5/5

Instructions-That paragraph above is all you need to play this game. The rulebook is as short as it needs to be. The game is an extremely simple to play game, so the rules don't have to be too difficult or cover too much territory. The extremely helpful thing included in this game is a cheat card for every player giving some quick iconography on how the different suit powers work. Honestly, this is a slick, simple rulebook that will get you playing in about 5 minutes even if youve never played a trick taking game before. 5/5

```
2015-08-21_1440178787

Execution-This game is a small game, but not a poorly put together one.  You ca
n see all the components here: http://youtu.be/dugtHKid-Ko (http://youtu.be/dug
tHKid-Ko) . The game is about a quarter the size and weight of most of my other
games, but that doesnt hurt its delivery.  The game comes with cardboard stande
e vaults, a deck of cards, and plastic diamonds.  What is here is well done and
beautiful.  The art is distinctive, but not distracting.  The diamonds are nice
plastic pieces that you want to collect.  Its a power-packed box.  5/5

Summary-Diamonds is the game I bring with me when I hang out with my family in
 Michigan.  Its got the simplicity of Hearts, but the depth I need in a great b
oard game.  It has great components and instructions.  My only real complaint i
s the theme, and the only reason I ding this game on theme is I play too many R
PGs, and I want theme in everything I play.  If a games story isn't the most im
portant thing to you, then this is an amazing, easy to play trick taking game t
hats a great game to add to any collection.  92.5%
```

I don't expect this to be good training data. Therefore, I am not going to use review above 750 words.

In [13]:
```
df_combined = df_combined[df_combined['wordCount'] < 750]
```

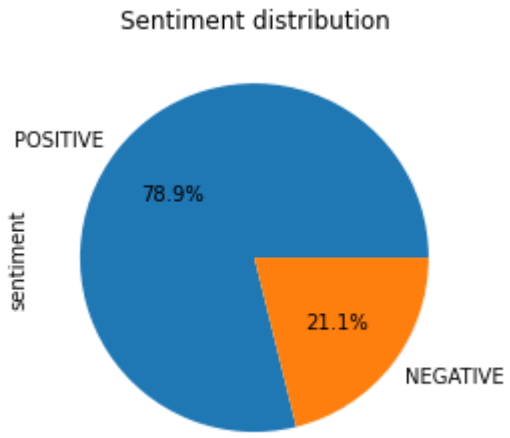
I also don't want to use reviews that are too short. Because these often consist of lists of things and not actual sentences. 6 words seems to me a nice number to have the boundary between sentences and listings

In [14]:
```
df_combined = df_combined[df_combined['wordCount'] > 6]
```

Sentiment distribution

How is the distribution between positive and negative reviews?

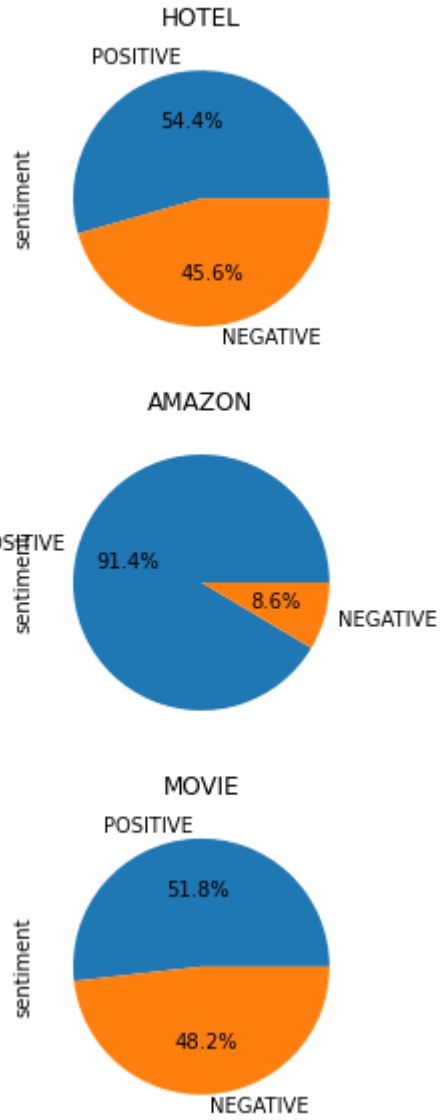
In [15]:
```
fig, ax = plt.subplots()
df_combined['sentiment'].value_counts().plot(ax=ax, kind='pie', autopct='%1.
plt.show()
```



There are many more positive reviews than negative ones. This does present a problem. It is wiser to train with an equal number of negative than positive reviews and keep the distribution even.

Let's also look at how the distribution by source is.

In [16]:
```
fig, ax = plt.subplots(3)
fig.set_figheight(10)
fig.set_figwidth(15)

df_combined[df_combined['source'] == 'HOTEL']['sentiment'].value_counts().pl
df_combined[df_combined['source'] == 'AMAZON']['sentiment'].value_counts().p
df_combined[df_combined['source'] == 'MOVIE']['sentiment'].value_counts().pl
plt.show()
```



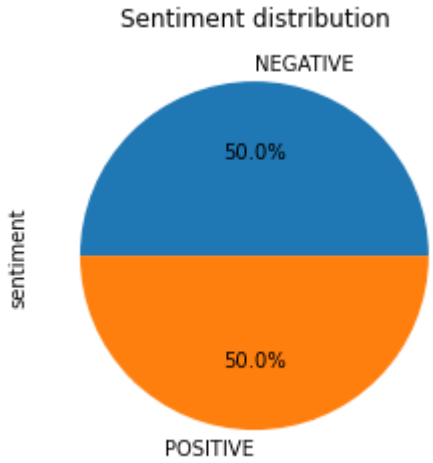
Let's make sure each source has an equal number of positive and negative reviews and then put them back together. In this way, the data on source and sentiment is simultaneously distributed as fairly as possible.

In [17]:
```
#create equalizer function
def sentiment_equalizer(df):
    #split by sentiment
    df_pos = df.loc[df['sentiment'] == 'POSITIVE']
    df_neg = df.loc[df['sentiment'] == 'NEGATIVE']

    #check length
    if len(df_pos) > len(df_neg):
        #equalise
        df_pos = df_pos[:len(df_neg)]
    else:
        #equalise
        df_neg = df_neg[:len(df_pos)]

    #combine
    return pd.concat([df_neg, df_pos])

```

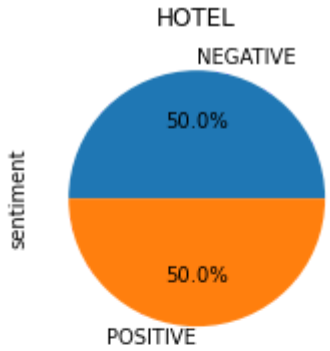
In [18]:
```
#apply equalizer function for each source
equal_hotel = sentiment_equalizer(df_combined[df_combined['source'] == 'HOTE
equal_amazon = sentiment_equalizer(df_combined[df_combined['source'] == 'AMA
equal_movie = sentiment_equalizer(df_combined[df_combined['source'] == 'MOVI

#combine
df_combined = pd.concat([equal_hotel, equal_amazon, equal_movie])
#shuffle
df_combined = df_combined.sample(frac=1, random_state=42)
#drop index
df_combined = df_combined.reset_index(drop=True)
```

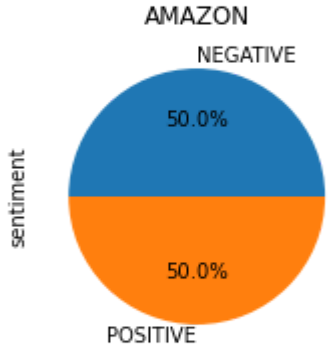
Let's check that the data is now better distributed by doing the same plots again.

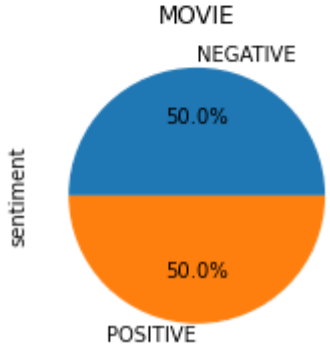
In [19]:
```
fig, ax = plt.subplots()
df_combined['sentiment'].value_counts().plot(ax=ax, kind='pie', autopct='%1.
plt.show()
```



In [20]:
```
fig, ax = plt.subplots(3)
fig.set_figheight(10)
fig.set_figwidth(15)

df_combined[df_combined['source'] == 'HOTEL']['sentiment'].value_counts().pl
df_combined[df_combined['source'] == 'AMAZON']['sentiment'].value_counts().p
df_combined[df_combined['source'] == 'MOVIE']['sentiment'].value_counts().pl
plt.show()
```







As you can see, sentiment is now fairly distributed by source.

Working with text data

According to sklearn's site on the page working with text data (https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html#extracting-features-from-text-files). Is bag of words the easiest way to recognize parts from text. To create a bag of words, CountVectorizer is used in the example.

Create bag of words

In [21]:
```
# import CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

Bag of words works in the following way: The input is a list of sentences. The output is a matrix with all the words that appear in these sentences. And for each sentence, a matrix with indexes referring to the words in the sentences.

This image explains it well. At the top you see a sentence. In the middle you see the matrix with indexes. And below the back of words.

In [22]:
```
# these are the input sentences
input_sentences = [
    "Here you have a test sentence",
    "This is another test sentence",
    "The word sentence is double in this sentence"
]

#create vectorizer
vect = CountVectorizer()

#create the matrix
matrix = vect.fit_transform(input_sentences)

#get the bag of words
bag_of_words = vect.get_feature_names()

#display
display(bag_of_words)
display(input_sentences)
display(matrix.toarray())
```

```
['another',
 'double',
 'have',
 'here',
 'in',
 'is',
 'sentence',
 'test',
 'the',
 'this',
 'word',
 'you']


['Here you have a test sentence',
 'This is another test sentence',
 'The word sentence is double in this sentence']


array([[0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0],
       [0, 1, 0, 0, 1, 1, 2, 0, 1, 1, 1, 0]], dtype=int64)
```

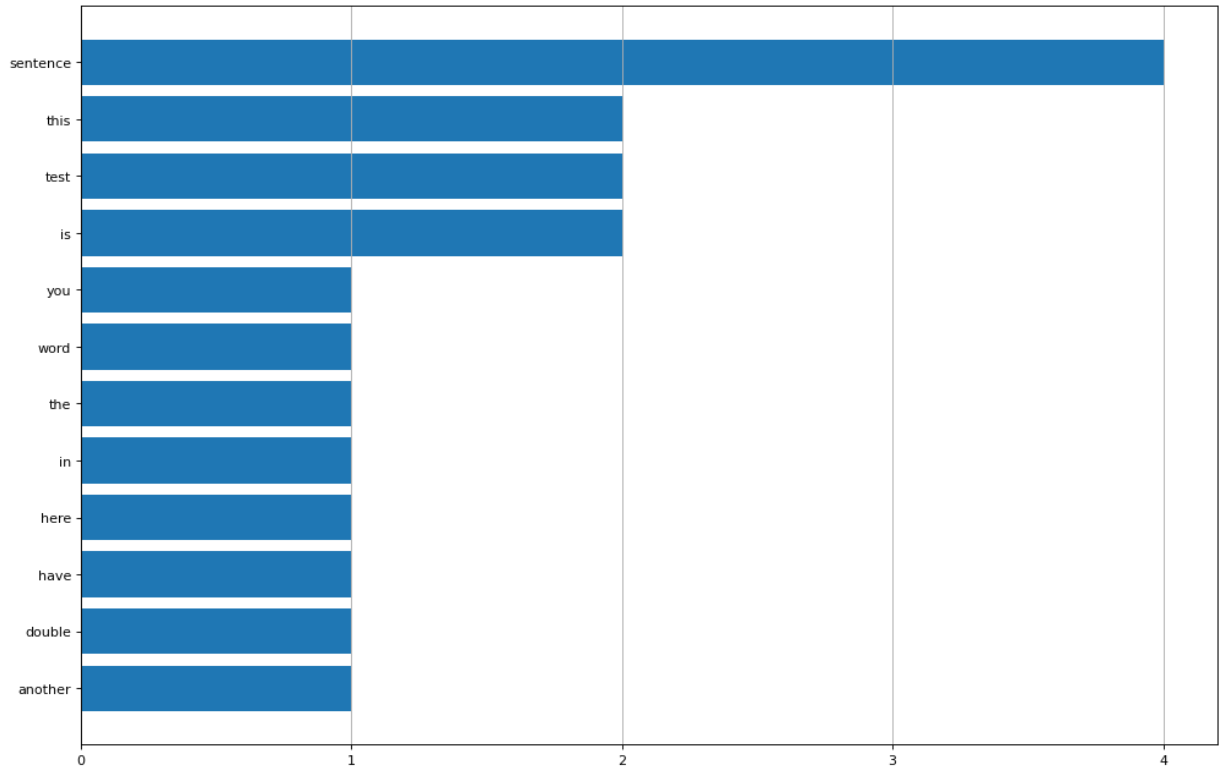
To get a nice overview of which words occur most often we are going to sum up the matrix.

In [23]:
```
matrix_total = np.asarray(matrix.sum(axis=0))[0]
matrix_total
```

Out[23]:
```
array([1, 1, 1, 1, 1, 2, 4, 2, 1, 2, 1, 1], dtype=int64)
```

We can use this summed matrix to create a nice graph that shows a good insight into word usage.

In [24]:
```
figure(figsize=(15, 10), dpi=80)

x = np.sort(matrix_total)
y = [x for _, x in sorted(zip(matrix_total,np.asarray(bag_of_words)))]

plt.barh(y, x)
plt.xticks(range(0,max(matrix_total)+1,1))
plt.grid(axis="x")
plt.show()
```



Real data

Now that it is clear how bag of words works, we can apply it to the real review texts.

In [25]:
```
%%time

#create vectorizer
vect = CountVectorizer()

#create the matrix
matrix = vect.fit_transform(df_combined['reviewText'])

#get the bag of words
bag_of_words = vect.get_feature_names()

#display
display(pd.Series(bag_of_words))
display(df_combined['reviewText'][:3])
display(matrix[:3].toarray())
```

```
0                        00
1                       000
2                     00000
3                    000000
4                  00000000
                ...
126451              zzzzzxx
126452              zzzzzzz
126453           zzzzzzzzzz
126454         zzzzzzzzzzzz
126455    zzzzzzzzzzzzzzzzz
Length: 126456, dtype: object
```

```
0                  Very tired hotel and noisy at night
1     Wasn t very keen in bedroom being in the base...
2     Front desk night manager charged my bill in d...
Name: reviewText, dtype: object
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
Wall time: 21.1 s
```

We can see a nice beginning here. But we also see a problem here: `zzzzzzzzzzzzzzzzzzzzz` is not really a useful word to train with

Filtering bag of words

To be more precise and create a more powerfull bag of words. I wrote a function that filters for numbers in words and checks if a word is in the dictionary.

In [26]:
```
#import library's
import enchant #dictionary library
import re #regex library

#create dictionary object
dictionary = enchant.Dict("en_US")

#create array to store filterd words
filterd_bag_of_words = []

for word in bag_of_words:
    #use regex to filer if word contains digets (=numeric)
    numeric = bool(re.search(r'\d', word))

    #use enchant to filter if word is in english dictionary
    is_a_word = dictionary.check(word)

    #if word is in dictionary and if nog numeric add to filterd array
    if (is_a_word and not(numeric)):
        filterd_bag_of_words.append(word)

#display filterd words
display(pd.Series(filterd_bag_of_words))
```

```
0             aa
1            aah
2       aardvark
3             ab
4            aba
          ...
44724     zoomed
44725    zooming
44726      zooms
44727       zoos
44728    zucchini
Length: 44729, dtype: object
```

Now that we have created a `filterd_bag_of_words` . We need to create a new vectorizer with and fill it with the filtered words.

In [27]:
```
#Create new vectorizer
vect = CountVectorizer()
#fit with filterd words
vect.fit(filterd_bag_of_words)
```

Out[27]:
```
CountVectorizer()
```

In [28]:
```
#check if words are indeed filterd
bag_of_words = vect.get_feature_names()
pd.Series(bag_of_words)
```

Out[28]:
```
0             aa
1            aah
2       aardvark
3             ab
4            aba
           ...
44724     zoomed
44725    zooming
44726      zooms
44727       zoos
44728    zucchini
Length: 44729, dtype: object
```

In [29]:
```
# make matrix for all reviews with new bag of words
matrix = vect.transform(df_combined['reviewText'])
```

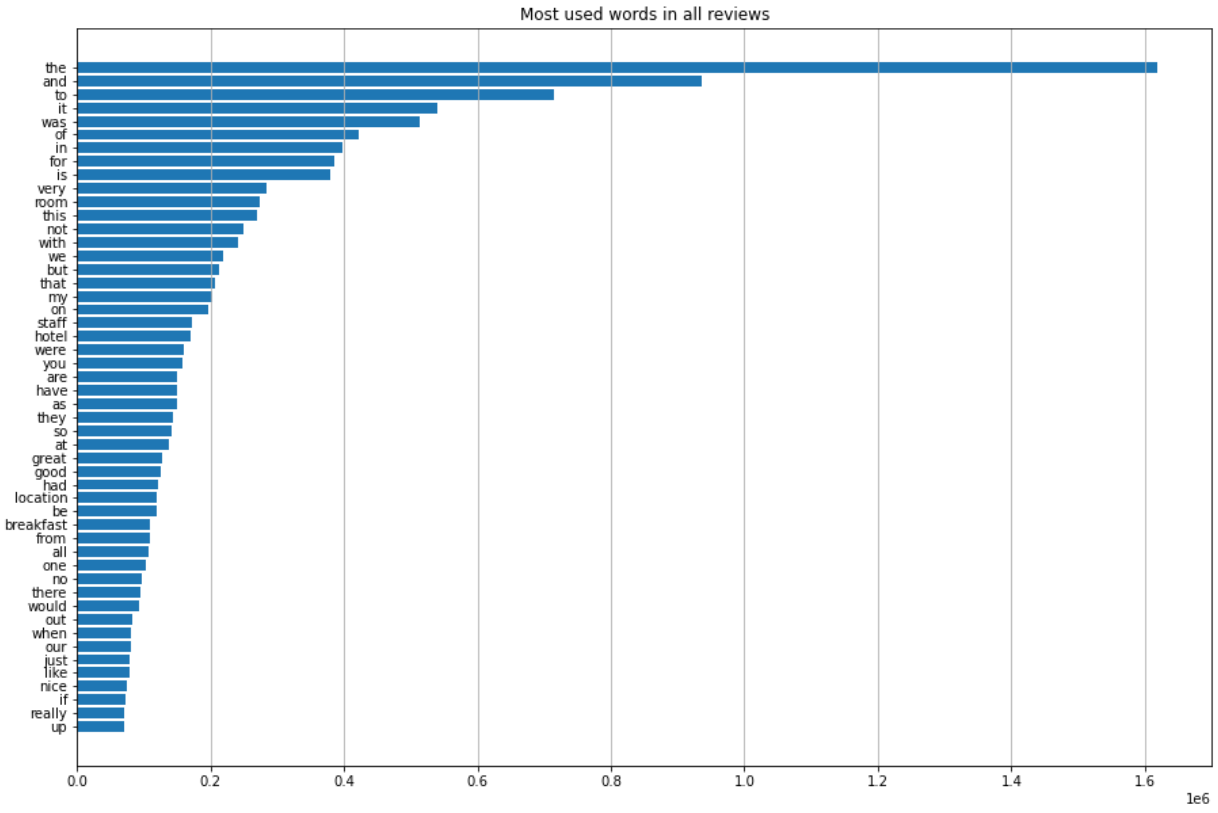
Bag of words analysis

Let's start by looking at what words are most commonly used.

In [30]:
```
#sum matrix
matrix_total = np.asarray(matrix.sum(axis=0))[0]
matrix_total
```

Out[30]:
```
array([596,   4,   1, ...,  41,   7,   2], dtype=int64)
```

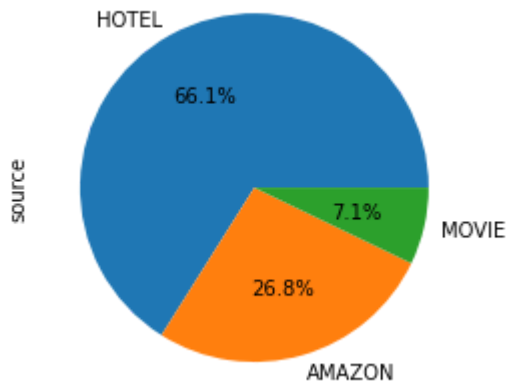
In [31]:
```
def plot_word_matrix(matrix, title):
    #sort matrix by highest number
    x = np.sort(matrix)
    #sort words by highest number in matrix
    y = [x for _, x in sorted(zip(matrix,np.asarray(bag_of_words)))]

    fig, ax = plt.subplots()

    #size
    fig.set_figheight(10)
    fig.set_figwidth(15)

    ax.set_title(title)

    #use top 50 words to make bar diagram
    ax.barh(y[-50:], x[-50:])
    #add x grid with steps of 2500
    ax.grid(axis="x")

#     plt.xticks(range(0,max(matrix),250000))

    plt.show()

plot_word_matrix(matrix_total,'Most used words in all reviews' )
```



We can see very clearly which words are often used here.

I notice that an awful lot of words seem to be related to hotel reviews. Although an earlier chart showed that most of the reviews came from the Amazon dataset. So how is that possible? In the meantime, of course, we have removed a lot of reviews. Let's look again at the source distribution.

In [32]:
```
fig, ax = plt.subplots()
df_combined['source'].value_counts().plot(ax=ax, kind='pie', autopct='%1.1f%
plt.show()
```



Knowing that (after editing the dataset) most of the reviews are indeed from the hotel dataset does make the bar digram clearer.

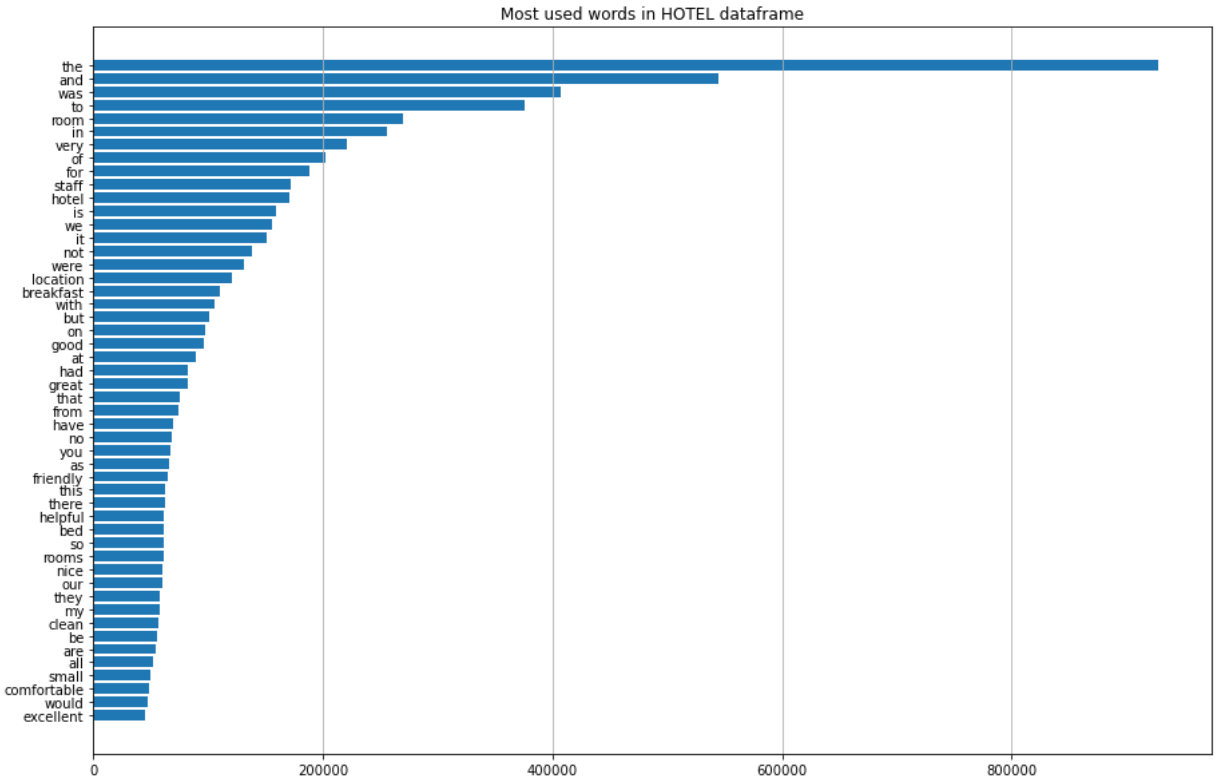
Seperated by source

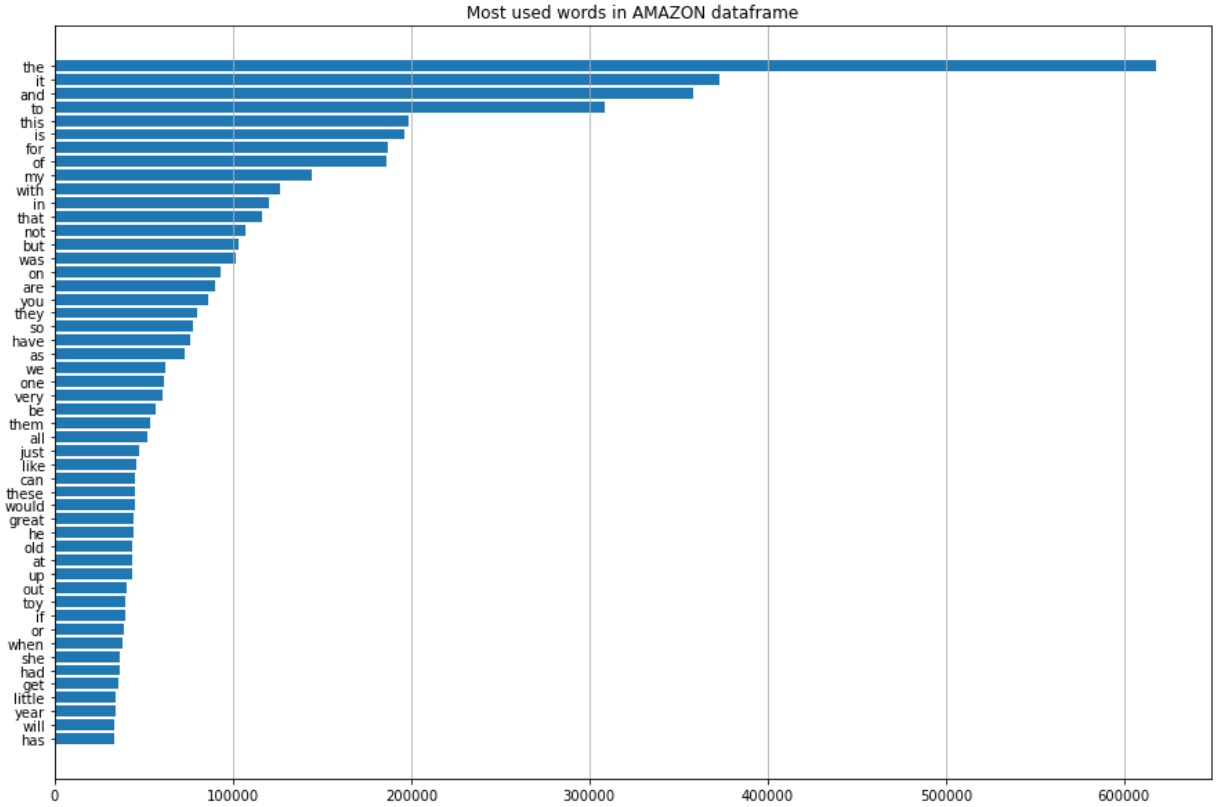
To see if the source has an impact on word use, we are going to make a similar graph. But then we separate the data by source.

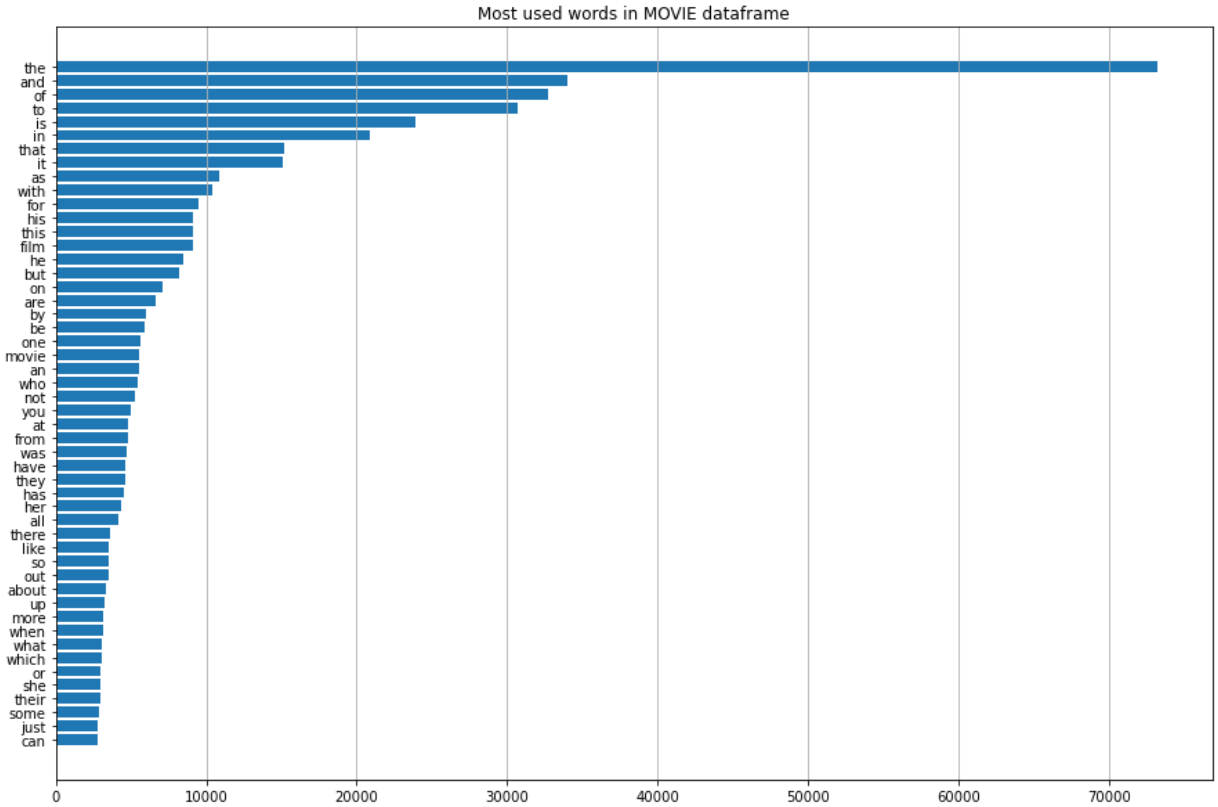
In [33]:
```
# make matrix per source
hotel_matrix = vect.transform(df_combined[df_combined['source'] == 'HOTEL'][
amazon_matrix = vect.transform(df_combined[df_combined['source'] == 'AMAZON'
movie_matrix = vect.transform(df_combined[df_combined['source'] == 'MOVIE'][
```

In [34]:
```
#sum matrixes
hotel_matrix_total = np.asarray(hotel_matrix.sum(axis=0))[0]
amazon_matrix_total = np.asarray(amazon_matrix.sum(axis=0))[0]
movie_matrix_total = np.asarray(movie_matrix.sum(axis=0))[0]
```

In [35]:
```
#plot all diagrams
plot_word_matrix(hotel_matrix_total,'Most used words in HOTEL dataframe')
plot_word_matrix(amazon_matrix_total,'Most used words in AMAZON dataframe')
plot_word_matrix(movie_matrix_total,'Most used words in MOVIE dataframe')
```




Most used words in AMAZON dataframe
the
it
and
to
this
is
for
of
my
with
in
that
not
but
was
on
are
you
they
so
have
as
we
one
very
be
them
all
just
like
can
these
would
great
he
old
at
up
out
toy
if
or
when
she
had
get
little
year
will
has
0
100000
200000
300000
400000
500000
600000


Most used words in MOVIE dataframe
the
and
of
to
is
in
that
it
as
with
for
his
this
film
he
but
on
are
by
be
one
movie
an
who
not
you
at
from
was
have
they
has
her
all
there
like
so
out
about
up
more
when
what
which
or
she
their
some
just
can
0
10000
20000
30000
40000
50000
60000
70000

You can see here that there is indeed a difference in word usage by source. Certain words like `movie`, `toy` and `hotel` are among the most used words in certain data frames. While at the same time these words do not appear at all in other data frames. This is understandable because a data frame with hotel reviews naturally talks more about hotels than a data frame with movie reviews.

To make an even better comparison, we are going to compare the most frequently used words of a data frame with how often it appears in the other data frames. The source distribution is not equal. So we need to normalize the word count to make a fair distribution.

In [36]:
```
#import library
from sklearn import preprocessing

# create normalise function
def normalise(arr):
    normalised = preprocessing.normalize([arr])
    normalised = normalised[0]
    normalised = normalised / max(normalised) * 1
    return normalised
```

In [37]:
```
# use normalise function to create normalised data
hotel_matrix_total_normalised = normalise(hotel_matrix_total)
amazon_matrix_total_normalised = normalise(amazon_matrix_total)
movie_matrix_total_normalised = normalise(movie_matrix_total)
```

Now that the normalized data is there. We can create a function that compares the most frequently used words, from a given dataset, to the other datasets.

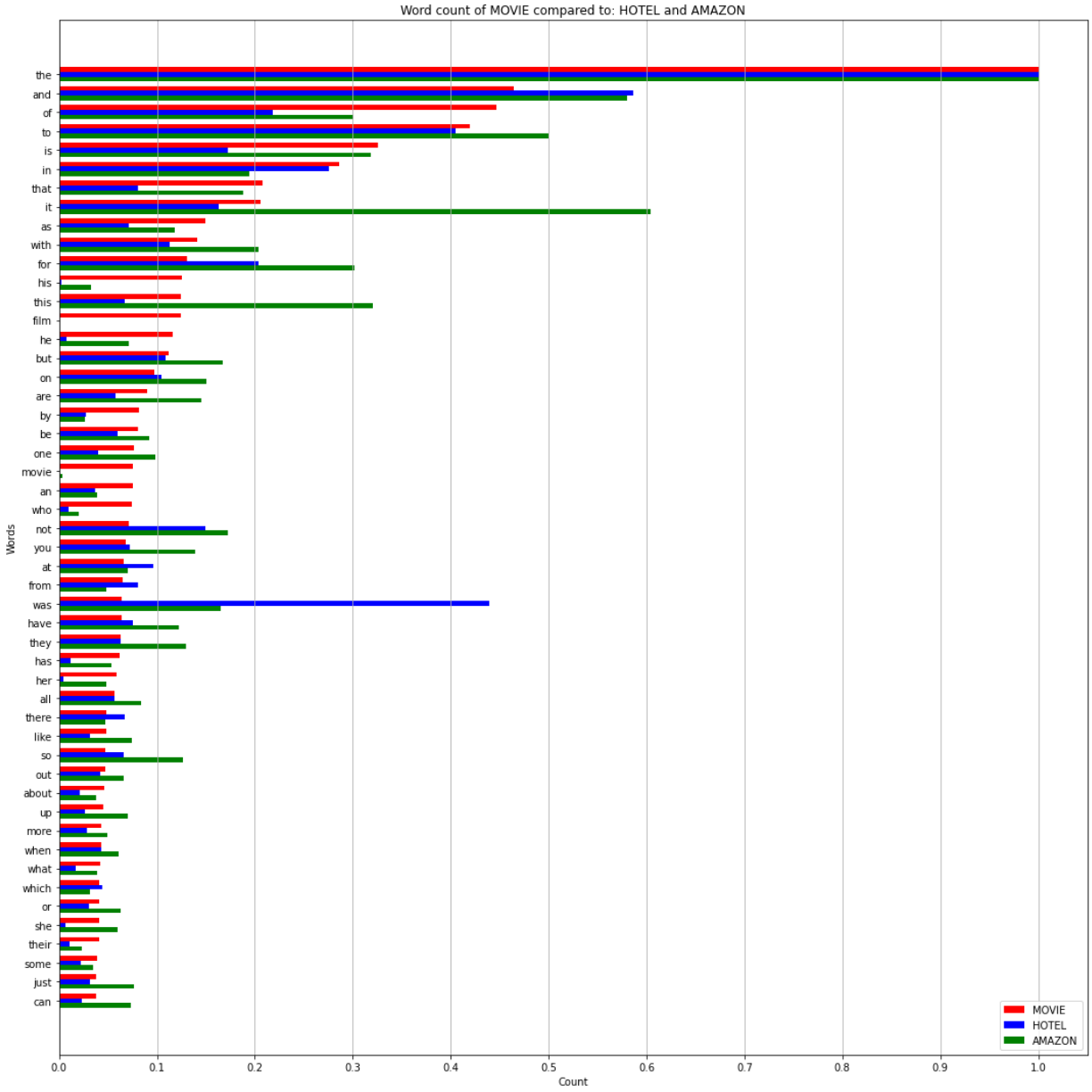
In [38]:
```
#create function
def plot_word_matrix_normalised(arr,arr_name, comp1, comp1_name, comp2, comp
    #Order labels by compared to arr
    labels = [x for _, x in sorted(zip(arr,np.asarray(bag_of_words)))][-50:]

    #Order arr, comp1 and comp2 counts by arr order
    arr_counts = [x for _, x in sorted(zip(arr,np.asarray(arr)))][-50:]
    comp1_counts = [x for _, x in sorted(zip(arr,np.asarray(comp1)))][-50:]
    comp2_counts = [x for _, x in sorted(zip(arr,np.asarray(comp2)))][-50:]

    x = np.arange(len(labels))  # the label locations
    width = 0.25  # the width of the bars

    fig, ax = plt.subplots()

    #size
    fig.set_figheight(15)
    fig.set_figwidth(15)

    #do hbar plots
    arr_bar   = ax.barh(x + width, arr_counts  , width, label=arr_name  , co
    comp1_bar = ax.barh(x        , comp1_counts, width, label=comp1_name, co
    comp2_bar = ax.barh(x - width, comp2_counts, width, label=comp2_name, co

    # Add some text for labels, title and custom x-axis tick labels, etc.
    ax.set_title('Word count of '+ arr_name + ' compared to: '+ comp1_name+
    ax.set_ylabel('Words')
    ax.set_yticklabels(labels)
    ax.set_xlabel('Count')
    ax.set_yticks(x)
    ax.legend()

    #add grid
    plt.xticks(np.arange(0, 1.1, 0.1))
    plt.grid(axis="x")

    fig.tight_layout()

    plt.show()

```

For each dataset, let's look only at the overrepresentation. Because underrepresentation is not visible in these graphs. This only becomes visible when you start comparing from another dataset.

Compare MOVIE

In [39]:
```
plot_word_matrix_normalised(
    movie_matrix_total_normalised,"MOVIE",
    hotel_matrix_total_normalised, "HOTEL",
    amazon_matrix_total_normalised, "AMAZON"
)
```

```
<ipython-input-38-884c12468d12>:28: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels(labels)
```



There are a number of words that grab my attention. I will try to briefly explain why these words

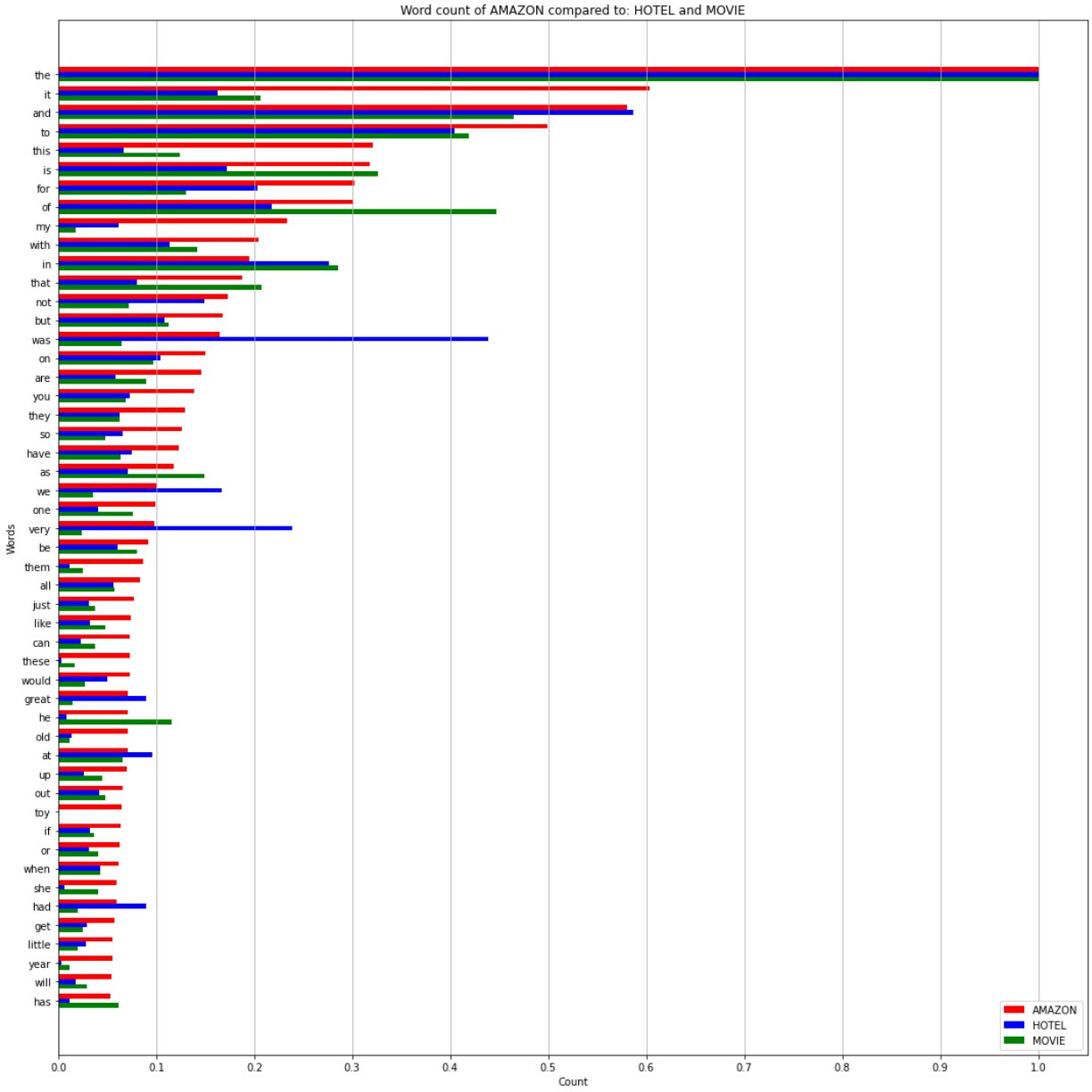
stand out to me.

- **film, movie**
  - These words refer directly to movies. So it also makes sense that they are over represented in this dataset.
- **about, who, his, he, her, she**
  - A movie review very often refers to the story or a character from the movie. These are all signal words that refer to a story or person/character. This is probably why these words are common in the movie dataset.
- **by**
  - The movie business is all about big names. The bigger the name the better the movie. So it makes sense that the creator/producer is often referenced right after a film title. The keyword "by" is therefore probably used so often.

Compare AMAZON

In [40]:
```
plot_word_matrix_normalised(
    amazon_matrix_total_normalised, "AMAZON",
    hotel_matrix_total_normalised, "HOTEL",
    movie_matrix_total_normalised,"MOVIE"
)
```

```
<ipython-input-38-884c12468d12>:28: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels(labels)
```



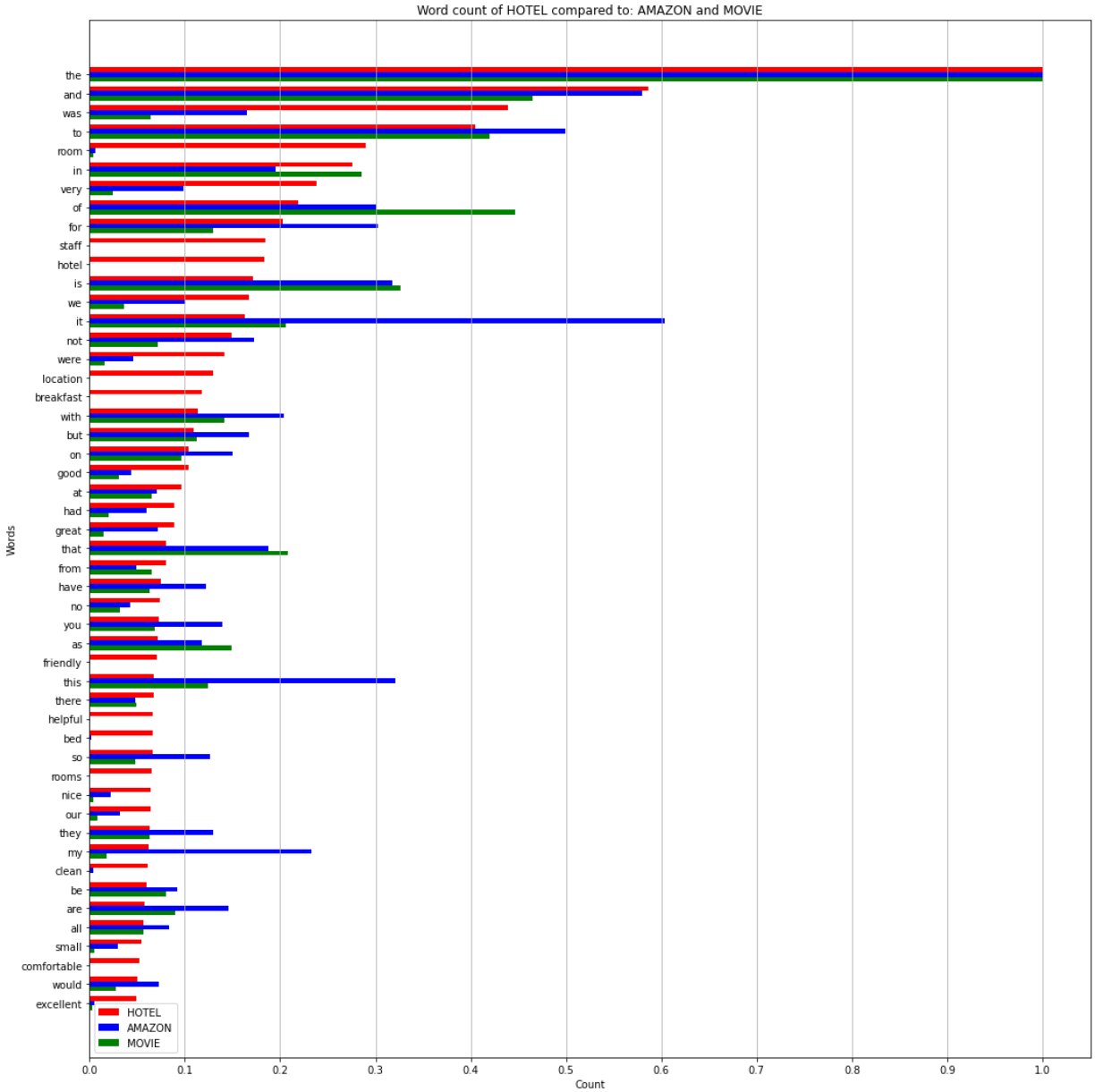
There are a number of words that grab my attention. I will try to briefly explain why these words stand out to me.

- **toy**
  - The Amazon dataset is from the subcategory "toys and games". So it makes sense that the word toy is very common in this dataset.
- **it, this, these**
  - These reviews are all about physical things rather than movies or hotels. "It, this and these" are therefore signal words referencing physical things.
- **for, my, year, old**
  - It probably happens often that people outline their situation at the beginning of their review so that readers can better understand the review. Since it is about toys and games, the following sentence will often occur.
    I bought this **for my** 4 **year old** child. and...
    I suspect that is why these words occur often.

Compare HOTEL

In [41]:
```
plot_word_matrix_normalised(
    hotel_matrix_total_normalised, "HOTEL",
    amazon_matrix_total_normalised, "AMAZON",
    movie_matrix_total_normalised,"MOVIE"
)
```

```
<ipython-input-38-884c12468d12>:28: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels(labels)
```



There are a number of words that grab my attention. I will try to briefly explain why these words stand out to me.

- **room, rooms, staff, friendly, helpfull, clean, comfortable, hotel, location, breakfast, bed**
  - These are words that refer to things or parts that you often find in hotels, and not in movies or toys.
- **was, had**
  - This may be because people often write the review at the request of the hotel. People often get an email from booking.com asking if they want to write a review. Maybe that's why people often write in the past tense in other reviews.
- **we**
  - It makes sense, somewhere, that plural is more often used when it comes to hotel reviews. After all, a hotel stay is something you experience more often in groups than on your own. This word therefore logically fits the hotel context.

So there are definitely differences in word usage by source. This gives very good insight into the data, but at the same time doesn't tell you very much about the sentiment of the text.

Seperated by sentiment

I want to make a similar chart but then I want to do the distribution not by source but by sentiment. To achieve this, we need to split the data by sentiment.

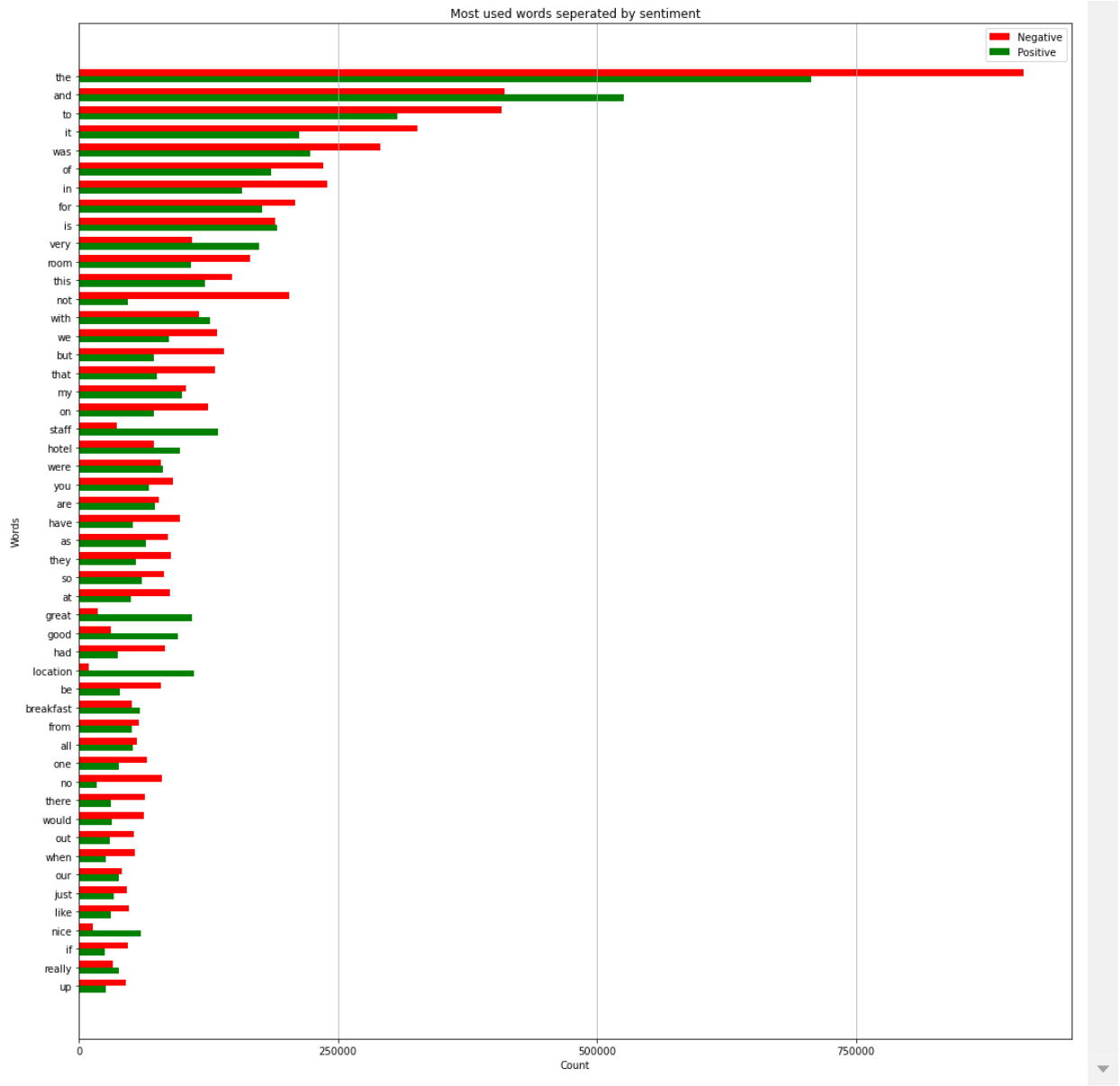
In [42]:
```
#make copy of review df
negative = df_combined
#get NEGATIVE
negative = negative[negative['sentiment'] == 'NEGATIVE']
#make matrix with excisting vectorizer (words)
neg_matrix = vect.transform(negative['reviewText'])
#sum matrix
neg_matrix = np.asarray(neg_matrix.sum(axis=0))[0]


#make copy of review df
positive = df_combined
#get POSITIVE
positive = positive[positive['sentiment'] == 'POSITIVE']
#make matrix with excisting vectorizer (words)
pos_matrix = vect.transform(positive['reviewText'])
#sum matrix
pos_matrix = np.asarray(pos_matrix.sum(axis=0))[0]
```

Now we can have a graph where both sentiments are used.

In [43]:

```
#sort words by highest number in total matrix
labels = [x for _, x in sorted(zip(matrix_total,np.asarray(bag_of_words)))][

#sort pos and neg counts by highest number in total matrix
pos_counts = [x for _, x in sorted(zip(matrix_total,np.asarray(pos_matrix)))
neg_counts = [x for _, x in sorted(zip(matrix_total,np.asarray(neg_matrix)))

x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots()

#size
fig.set_figheight(15)
fig.set_figwidth(15)


negative = ax.barh(x + width/2, neg_counts, width, label='Negative', color=
positive = ax.barh(x - width/2, pos_counts, width, label='Positive', color=

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_title('Most used words seperated by sentiment')
ax.set_ylabel('Words')
ax.set_yticklabels(labels)
ax.set_xlabel('Count')
ax.set_yticks(x)
ax.legend()


plt.xticks(range(0,max(neg_counts),250000))
plt.grid(axis="x")

fig.tight_layout()
plt.show()
```

```
<ipython-input-43-8bf4fe2a88e3>:24: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_yticklabels(labels)
```



Difference in occurrences

You can clearly see here that there are clear differences in occurrences. To draw any real conclusions we need to focus on the number of differences in occurrences. So let's make a neat graph focussing on the differences.

In [44]:
```
# function that calculates the difference
def diff_calculator(pos, neg):
    diff_arr = []
    for ind in range(len((pos))):
        diff = pos[ind] - neg[ind]
        diff_arr.append(diff)

        #print first 5 elements to demonstrate what de function does.
        if ind < 5:
            print('pos:', pos[ind], 'neg:', neg[ind], 'diff:', diff)
    return diff_arr

#use function
diff_pos_neg = diff_calculator(pos_matrix,neg_matrix)
```

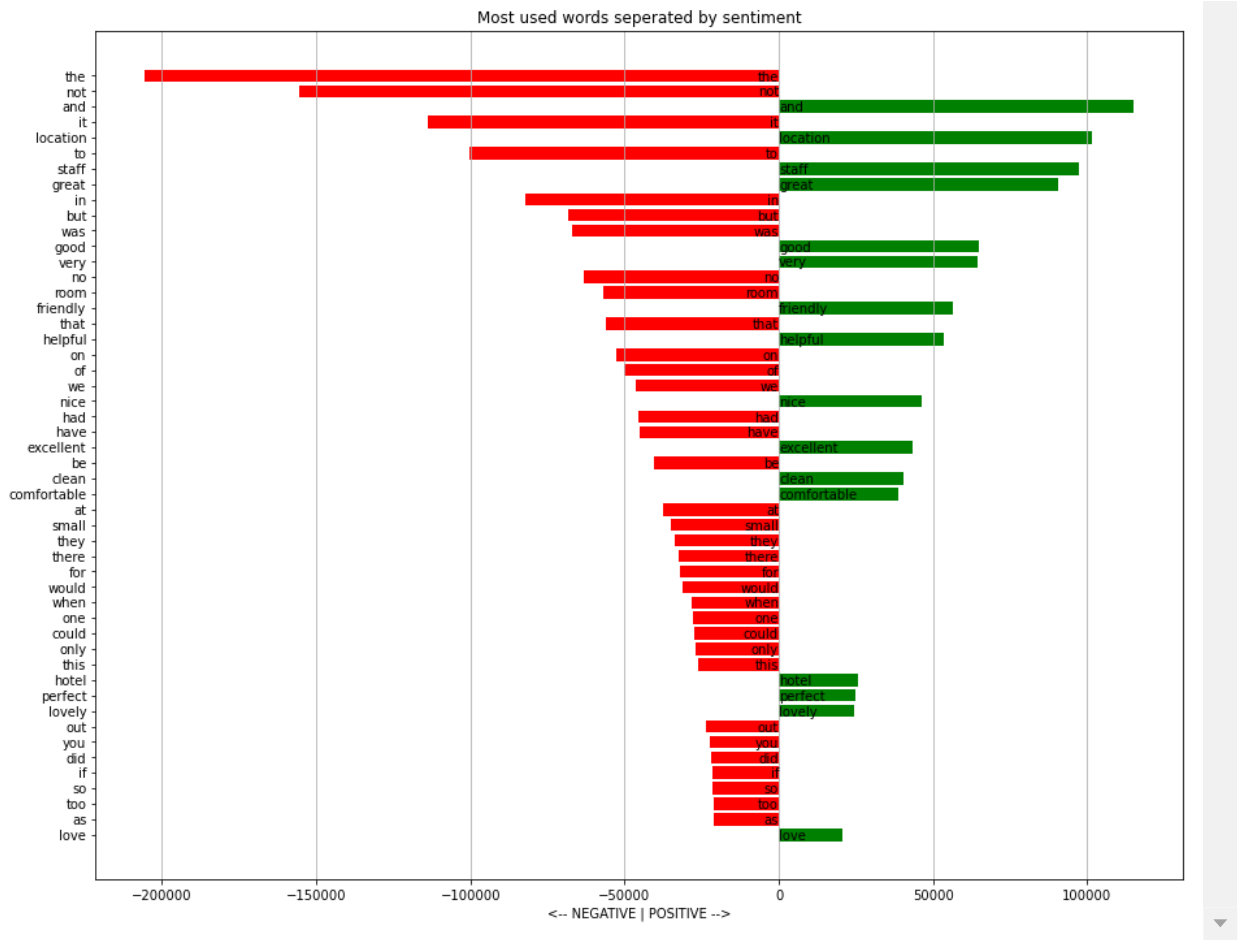
```
pos: 321 neg: 275 diff: 46
pos: 3 neg: 1 diff: 2
pos: 0 neg: 1 diff: -1
pos: 28 neg: 25 diff: 3
pos: 2 neg: 1 diff: 1
```

In [45]:
```
#Sort lables by absolute (positive) diff values
diff_labels = [x for _, x in sorted(zip(map(abs,diff_pos_neg),np.asarray(bag
#Sort diff values by absolute (positive) diff values
diff = [x for _, x in sorted(zip(map(abs,diff_pos_neg),np.asarray(diff_pos_n
```

In [46]:

```
#make figure bigger
figure(figsize=(15, 10), dpi=80)

fig, ax = plt.subplots()

#size
fig.set_figheight(12)
fig.set_figwidth(15)

ax.set_title('Most used words seperated by sentiment')

positive = pd.Series(diff)[-50:] > 0

#use top 50 words to make bar diagram
ax.barh(diff_labels[-50:], diff[-50:], color=positive.map({True: 'g', False:
#add x grid
ax.grid(axis="x")

#add labes on x = 0 for readability
for ind in range(50):
    if np.array(positive)[ind]:
        ha = 'left'
    else:
        ha = 'right'
    ax.text(0, -.5+ind, diff_labels[-50:][ind],
            ha=ha, va='bottom')

#add label that shows negative vs positive
plt.xlabel("<-- NEGATIVE | POSITIVE -->")
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



Word use explanation

In this chart you can clearly see the difference in word usage. The word "the" was used almost 300,000 times more often in negative reviews than in positive reviews. This example is probably coincidental. But there are also words that stand out:

- **great, good, nice, excellent, perfect, lovely**
  - These are all words that belong to liking something. So it is also natural that these words were used more in **positive** reviews.
- **location, staff**
  - These words probably come from the hotel review dataset. However, it is noticeable that these components are more often reviewed positively than negatively.
- **friendly, helpful, clean, comfortable**
  - These are also likely to be words from the hotel review dataset. So apparently reviewers think it's very important that things are clean and comfortable. And people are friendly and helpful.

- **and, but**
  - These words are both linking words. They do mean something different. **And** is a word that indicates **addition**, while **but** signals **contrast**. So we can assume that **positive** text **lists** things more often. And **negative** reviews more often use a **contradiction**.
- **not**
  - not is a word often used in negative reviews. It is a word used in opposites. This is another confirmation that **contradictions** are used a lot in **negative** reviews.
- **was, had**

- These words are both written in past tense. It is likely that **negative** text was written in **past tense** more often than not.

Conclusion

We learned a lot about all the datasets. In particular, the word usage for each dataset provided many insights. In fact, the word usage per dataset varies greatly. Certain words that fit the context of the data are used more often. We also know that hotel reviews are more often written in plural, because it is something you experience with several people.

We learned about many signal words for positive and negative reviews. We also learned that in positive reviews, things are often listed, and they contain more words on average. And in negative reviews, contradictions are more often used. Also, negative text is more often written in past tense.

I think at this point I understand the data well. And I'm confident that I can proceed to the machine learning phase.

Save dataframe

To use the filterd dataframe, the vectoriser, the matrix and the bag of words, for the machine learning phase. I' m going to use a library that allows you to save objects as a file.

In [47]:
```
#import pickle to store objects to file
import pickle

object_array = [vect, matrix, bag_of_words, df_combined]
with open('obj/object_array.pkl', 'wb') as f:
    pickle.dump(object_array, f)
```

Modeling

Project

Several sites allow you to post a comment or opinion about a product, place, or event. Think of Facebook, Instagram, Youtube, or Reddit. However, on not all of these sites, you can see at a glance how many of these comments are written with a positive or negative mindset.

For many people or companies, it is useful to know what percentage of the reactions are positive or negative. You can't always tell this by looking at a Like system. The main question you should be able to answer with the help of this project is; Is a piece of text positive or negative?

Document goal

In this paper, I want to find and train the best model to classify text sentiment. I am going to look at how accurate the models are and substantiate why models achieve these scores. Then I will choose the best one to use for the deployment phase.

To get started, we need some libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pickle
```

We need the saved objects from the previous file. We can import these using pickle.

In [2]:
```
with open('obj/object_array.pkl', 'rb') as f:
    vect, matrix, bag_of_words, df_combined = pickle.load(f)
```

Splitting data

Now we can almost start training! But before we get to that point we need to split the data into test and training data. Because I want to try many different things quickly at first, we only use a small portion of the data.

In [3]:
```
from sklearn.model_selection import train_test_split

def split_data(sample_size):
    matrix_train, matrix_test, df_train, df_test =  train_test_split(matrix[
                                                                     df_comb
                                                                     test_si
                                                                     random_


    sentiment_train = np.array(df_train['sentiment'])
    review_train = np.array(df_train['reviewText'])
    source_train = np.array(df_train['source'])

    sentiment_test = np.array(df_test['sentiment'])
    review_test = np.array(df_test['reviewText'])
    source_test = np.array(df_test['source'])

    return matrix_train, matrix_test, sentiment_train, sentiment_test, revie

N = 15000 #sample out of 815804 ~1,8%
matrix_train, matrix_test, sentiment_train, sentiment_test, review_test, sou
```

Naive bayes

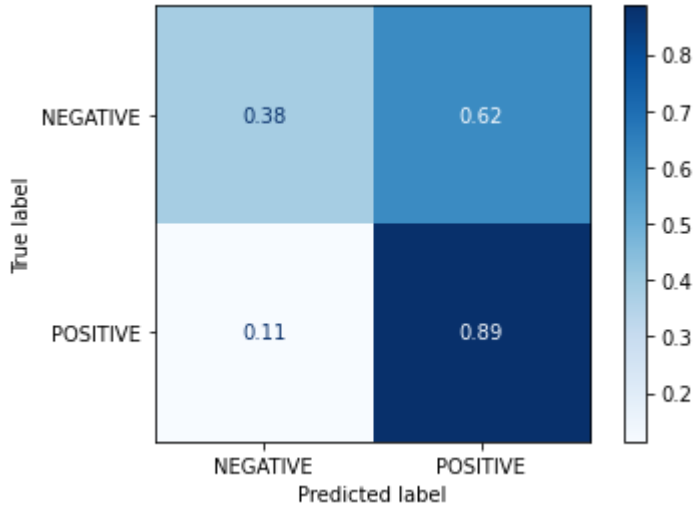
I want to start by looking at Naive bayes. This is a classification technology that should work extremely well with text data. Let's just try a few different types and then see which ones score well.
To properly see how a classifier scores, we first create a function that shows a classification report and confusion matrix.

In [4]:
```
#import metrics components
from sklearn.metrics import classification_report
from sklearn.metrics import plot_confusion_matrix

#create function that prints reports
def print_report(clf, to_array):

    if to_array:
        pred = clf.predict(matrix_test.toarray())
    else:
        pred = clf.predict(matrix_test)


    print(f"Classification report for classifier {clf}:\n"
    f"{classification_report(sentiment_test, pred)}\n")

    plot_confusion_matrix(clf, matrix_test.toarray(), sentiment_test,
                          cmap=plt.cm.Blues,
                          normalize='true'
                      )
```

Gaussian Naive bayes

In [5]:
```
%%time
from sklearn.naive_bayes import GaussianNB
clf_gnb = GaussianNB()
clf_gnb.fit(matrix_train.toarray(), sentiment_train)
print_report(clf_gnb, True)
```

```
Classification report for classifier GaussianNB():
              precision    recall  f1-score   support

    NEGATIVE       0.77      0.38      0.51      2469
    POSITIVE       0.59      0.89      0.71      2481

    accuracy                           0.64      4950
   macro avg       0.68      0.63      0.61      4950
weighted avg       0.68      0.64      0.61      4950


Wall time: 17.8 s
```

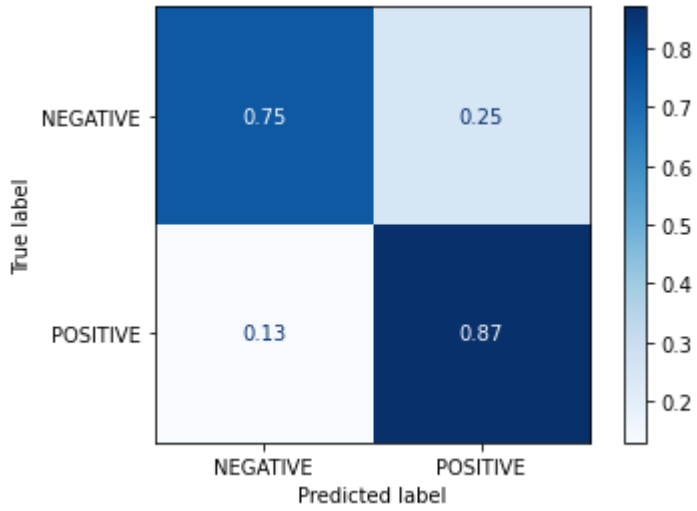


We see here an accuracy of 64% this is ansight not very high. We also see that there are 6 times as many false positives than false negatives. In fact, there are more false positives than true negatives. I am not very satisfied with this result yet.

Bernoulli Naive bayes

In [6]:
```
%%time
from sklearn.naive_bayes import BernoulliNB
clf_bnb = BernoulliNB()
clf_bnb.fit(matrix_train.toarray(), sentiment_train)
print_report(clf_bnb, True)
```

```
Classification report for classifier BernoulliNB():
              precision    recall  f1-score   support

    NEGATIVE       0.85      0.75      0.80      2469
    POSITIVE       0.78      0.87      0.82      2481

    accuracy                           0.81      4950
   macro avg       0.82      0.81      0.81      4950
weighted avg       0.82      0.81      0.81      4950


Wall time: 24.6 s
```

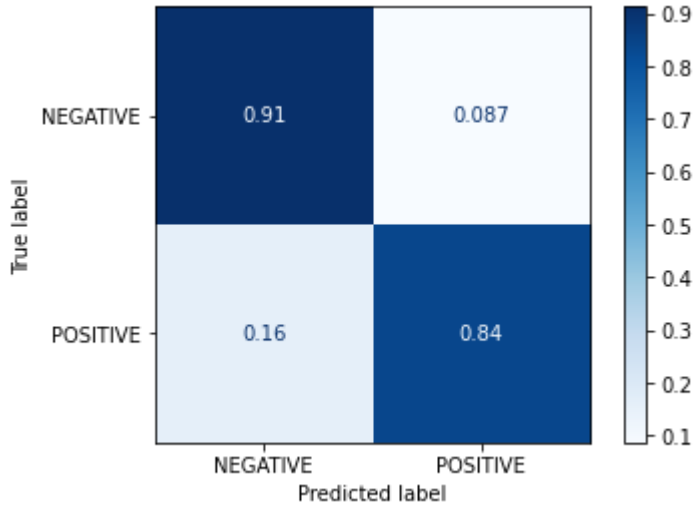


Here the accuacy score is already a lot higher; 81% is quite high. We do see twice as many false positives than false negatives. This is better than 6 times as many. But it is still a big difference.

Mulinomial Naive bayes

In [7]:
```
%%time
from sklearn.naive_bayes import MultinomialNB
clf_mnb = MultinomialNB()
clf_mnb.fit(matrix_train.toarray(), sentiment_train)
print_report(clf_mnb, True)
```

```
Classification report for classifier MultinomialNB():
              precision    recall  f1-score   support

    NEGATIVE       0.85      0.91      0.88      2469
    POSITIVE       0.91      0.84      0.87      2481

    accuracy                           0.88      4950
   macro avg       0.88      0.88      0.88      4950
weighted avg       0.88      0.88      0.88      4950


Wall time: 18.2 s
```

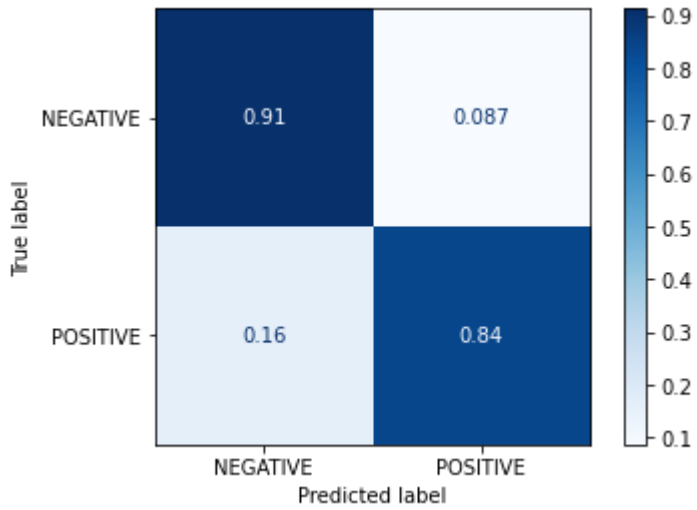


88% is a result we can be very satisfied with! Also, there is not much difference between the false positives and false negatives. It is worth noting that with Bernoulli and Gaussian, there are more false positives, while with Mulinominal there are more false negatives.

Complement Naive bayes

In [8]:
```
%%time
from sklearn.naive_bayes import ComplementNB
clf_cnb = ComplementNB()
clf_cnb.fit(matrix_train.toarray(), sentiment_train)
print_report(clf_cnb, True)
```

```
Classification report for classifier ComplementNB():
              precision    recall  f1-score   support

    NEGATIVE       0.85      0.91      0.88      2469
    POSITIVE       0.91      0.84      0.87      2481

    accuracy                           0.88      4950
   macro avg       0.88      0.88      0.88      4950
weighted avg       0.88      0.88      0.88      4950


Wall time: 17.9 s
```



These are exactly the same results as for Mulinomial. I'm very curious to know if just the score happens to be the same, or if actually the same is classified. To find out, I want to compare the misclassified results.

In [9]:
```
# get predicted values
pred_mnb = clf_mnb.predict(matrix_test.toarray())
pred_cnb = clf_cnb.predict(matrix_test.toarray())
```

In [10]:
```
# get misclassified indexes by comparing sentiment_test to predicted values
misclassified_mnb = np.where(sentiment_test != pred_mnb)
misclassified_cnb = np.where(sentiment_test != pred_cnb)
```

In [11]:
```
#print first 100 values to look at misclassified indexes
print(misclassified_mnb[0][:100])
print(misclassified_cnb[0][:100])

# get indexes where Mulinomial misclassification is not the same as Compleme
np.where(misclassified_mnb[0] != misclassified_cnb[0])
```

```
[  0  11  23  28  36  49  54  63  67  68  76  91  97 106 114 119 133 135
 153 166 176 199 209 223 224 228 239 255 259 266 271 288 293 301 306 314
 316 323 333 340 346 370 373 396 410 418 423 432 448 453 456 464 477 482
 500 510 521 530 533 542 546 547 548 549 555 573 585 600 612 621 633 645
 665 666 691 702 705 709 714 719 723 729 732 739 748 751 759 764 774 782
 812 814 817 821 829 837 844 862 884 889]
[  0  11  23  28  36  49  54  63  67  68  76  91  97 106 114 119 133 135
 153 166 176 199 209 223 224 228 239 255 259 266 271 288 293 301 306 314
 316 323 333 340 346 370 373 396 410 418 423 432 448 453 456 464 477 482
 500 510 521 530 533 542 546 547 548 549 555 573 585 600 612 621 633 645
 665 666 691 702 705 709 714 719 723 729 732 739 748 751 759 764 774 782
 812 814 817 821 829 837 844 862 884 889]
```

Out[11]:
```
(array([], dtype=int64),)
```

If you look at the first 100 numbers, you can also see that they already appear to be the same. And when we actually start comparing and start checking where the differences are, we get back an empty array. This means that there is no difference between the two classifiers.

Naive bayes conclusion

Because Complemented and Mulinominal seem to score equally well. We can conclude that it does not matter which model we would choose. Therefore, we go for the fastest train option. That is Complemented naive bayes.

SVM

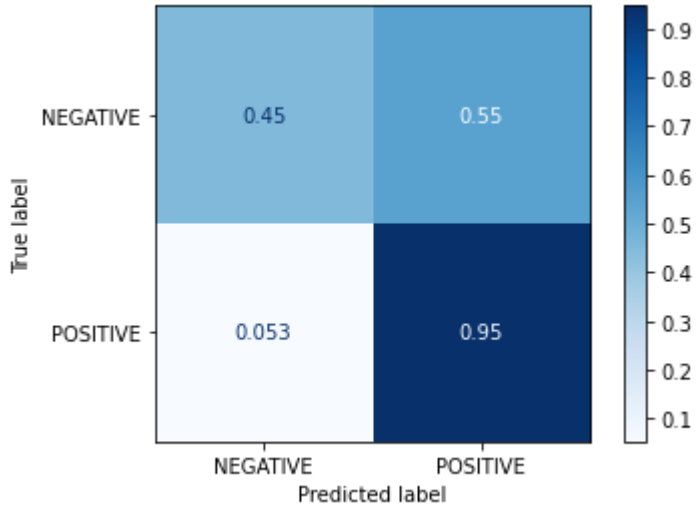
Before we put all our hopes on naive bayes I would also like to look at svm. There are very many good and interesting options that will most likely give good results. Let's start to test every kind of kernel.

In [12]:
```
#import svm
from sklearn import svm
```

Poly

In [13]:
```
%%time
clf_ply = svm.SVC(kernel='poly')
clf_ply.fit(matrix_train, sentiment_train)
print_report(clf_ply, False)
```

```
Classification report for classifier SVC(kernel='poly'):
              precision    recall  f1-score   support

    NEGATIVE       0.89      0.45      0.60      2469
    POSITIVE       0.63      0.95      0.76      2481

    accuracy                           0.70      4950
   macro avg       0.76      0.70      0.68      4950
weighted avg       0.76      0.70      0.68      4950


Wall time: 32 s
```

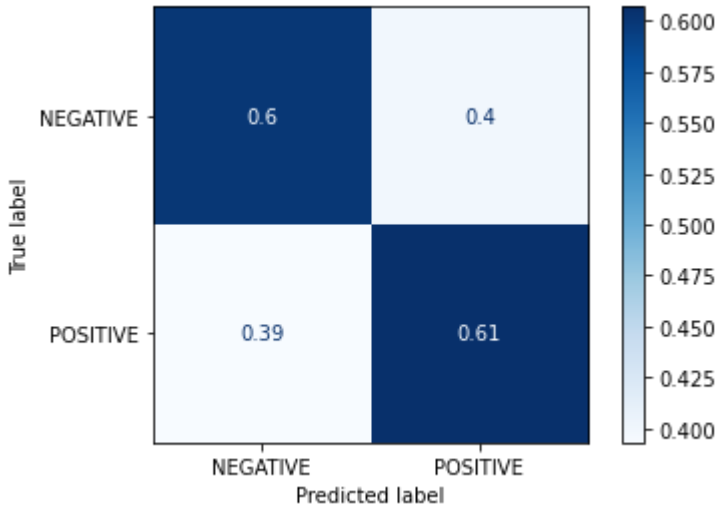


The results of this kernel are very disappointing. At first, an acccuracy of 70% seems acceptable. But if you look at the confusion matrix you can see that there are more false positives, than true negatives. This is really a problem if you would use this model for real world examples. A lot of sentences will be seen as positive when they are not.

Sigmoid

In [14]:
```
%%time
clf_sig = svm.SVC(kernel='sigmoid')
clf_sig.fit(matrix_train, sentiment_train)
print_report(clf_sig, False)
```

```
Classification report for classifier SVC(kernel='sigmoid'):
              precision    recall  f1-score   support

    NEGATIVE       0.60      0.60      0.60      2469
    POSITIVE       0.60      0.61      0.61      2481

    accuracy                           0.60      4950
   macro avg       0.60      0.60      0.60      4950
weighted avg       0.60      0.60      0.60      4950


Wall time: 19.5 s
```

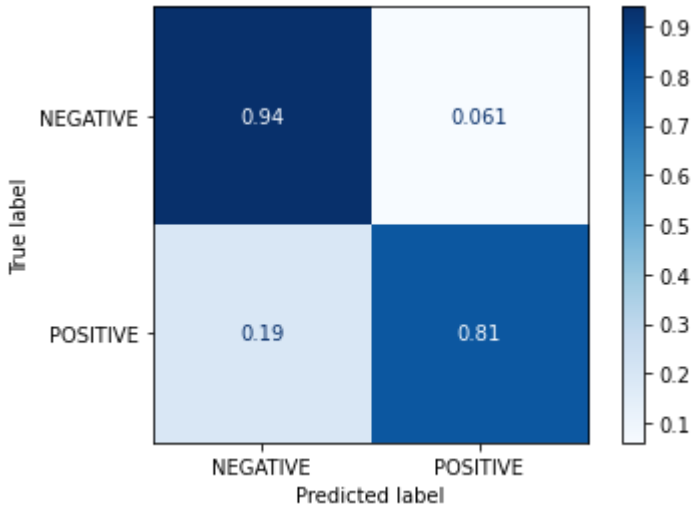


An accuracy of 60% is not very high. However, it is interesting to note that the distribution of misclassified is almost the same. Both 40%.

RBF

In [15]:
```
%%time
clf_rbf = svm.SVC(kernel='rbf')
clf_rbf.fit(matrix_train, sentiment_train)
print_report(clf_rbf, False)
```

```
Classification report for classifier SVC():
              precision    recall  f1-score   support

    NEGATIVE       0.83      0.94      0.88      2469
    POSITIVE       0.93      0.81      0.86      2481

    accuracy                           0.87      4950
   macro avg       0.88      0.87      0.87      4950
weighted avg       0.88      0.87      0.87      4950


Wall time: 25.1 s
```

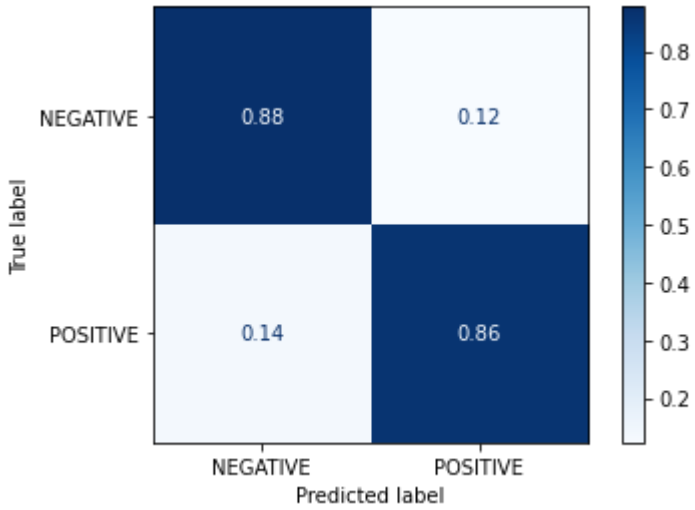


This is a very neat score. 87% we can be very happy with. However, this model is extra sensitive to negative results. We may be able to overcome this later with gamma and c parameters.

Linear

In [16]:
```
%%time
clf_lin = svm.SVC(kernel='linear')
clf_lin.fit(matrix_train, sentiment_train)
print_report(clf_lin, False)
```

```
Classification report for classifier SVC(kernel='linear'):
              precision    recall  f1-score   support

    NEGATIVE       0.86      0.88      0.87      2469
    POSITIVE       0.88      0.86      0.87      2481

    accuracy                           0.87      4950
   macro avg       0.87      0.87      0.87      4950
weighted avg       0.87      0.87      0.87      4950


Wall time: 18.8 s
```



Again, the accuracy is very high (87%). It is noticeable that when you look at the confussion matrix, and compare it to that of RBF, the distribution of false labled data is more equal.

Grid search

Both with linear and rbf we achieved reasonably high results (both 87%). This can probably be increased a bit when playing with gamma and c parameters, especially with rbf. To automatically find the best values for this we use grid search.

In [17]:
```
%%time
from sklearn.model_selection import GridSearchCV

parameters = {
    'kernel': ('linear', 'rbf'),
    'C': np.logspace(-2, 6, 5).tolist(),
    'gamma': np.logspace(-9, 3, 13).tolist() + [ 'auto', 'scale']
    }

print("Kernels: ", parameters['kernel'])
print("C: ", parameters['C'])
print("Gamma: ", parameters['gamma'])

svc = svm.SVC()
clf_gs = GridSearchCV(svc, parameters)
clf_gs.fit(matrix_train, sentiment_train)
```

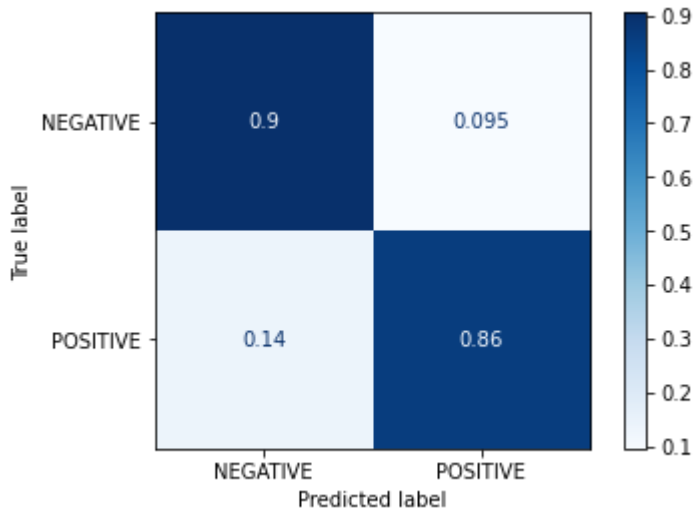
```
Kernels:  ('linear', 'rbf')
C:  [0.01, 1.0, 100.0, 10000.0, 1000000.0]
Gamma:  [1e-09, 1e-08, 1e-07, 1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1, 1.0, 10.
0, 100.0, 1000.0, 'auto', 'scale']
Wall time: 2h 33min 38s
```

Out[17]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.01, 1.0, 100.0, 10000.0, 1000000.0],
                         'gamma': [1e-09, 1e-08, 1e-07, 1e-06, 1e-05, 0.0001,
                                   0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0,
                                   'auto', 'scale'],
                         'kernel': ('linear', 'rbf')})
```

This took quite a while (over 2 hours). But let's see what results we get from this.

In [18]:
```
1 print_report(clf_gs, False)
```

```
Classification report for classifier GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.01, 1.0, 100.0, 10000.0, 1000000.0],
                         'gamma': [1e-09, 1e-08, 1e-07, 1e-06, 1e-05, 0.0001,
                                   0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0,
                                   'auto', 'scale'],
                         'kernel': ('linear', 'rbf')}):
              precision    recall  f1-score   support

    NEGATIVE       0.87      0.90      0.89      2469
    POSITIVE       0.90      0.86      0.88      2481

    accuracy                           0.88      4950
   macro avg       0.88      0.88      0.88      4950
weighted avg       0.88      0.88      0.88      4950
```



An accuracy of 88% is very good! This is exactly the same as the naive bayes method. We do see that the model is a little bit oversensitive to negative reviews. But the difference is negligible.
But what are the best parameters found by the grid search?

In [19]:
```
1 clf_gs.best_params_
```

Out[19]:
```
{'C': 100.0, 'gamma': 0.001, 'kernel': 'rbf'}
```

Svm Conclusion

The rbf kernel with a gamma of 0.001 and a C of 100 we achieve the best results.

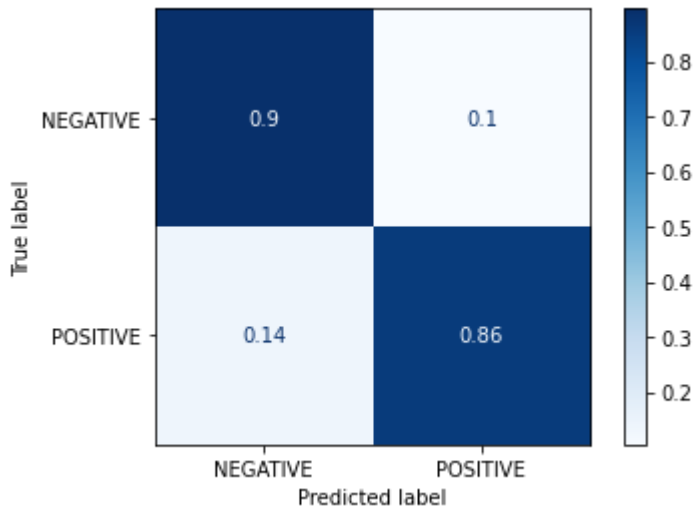
More training data

We have learned that, Complement Naive bayes and SVM rbf, both present exactly equally well. Before making a final choice, we will re-train both with more data to hopefully come up with an even better model.

In [20]:
```
N = 100000 #sample out of 815804 ~12,3%
matrix_train, matrix_test, sentiment_train, sentiment_test, review_test, sou
```

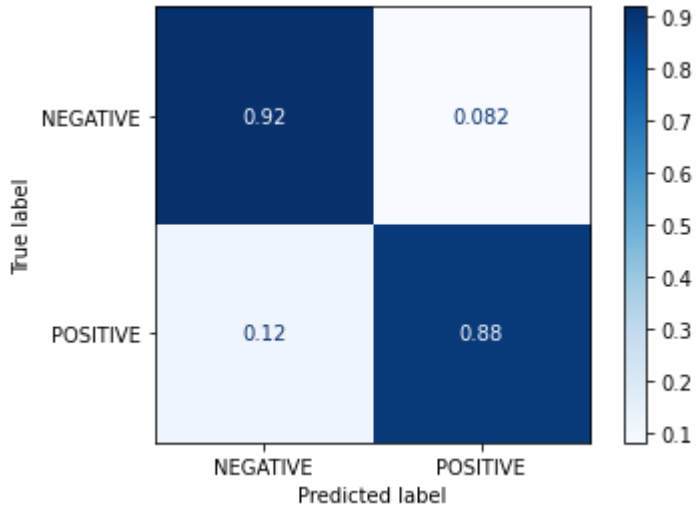
In [21]:
```
%%time
clf_cnb = ComplementNB()
clf_cnb.fit(matrix_train.toarray(), sentiment_train)
print_report(clf_cnb, True)
```

```
Classification report for classifier ComplementNB():
              precision    recall  f1-score   support

    NEGATIVE       0.87      0.90      0.88     16589
    POSITIVE       0.89      0.86      0.87     16411

    accuracy                           0.88     33000
   macro avg       0.88      0.88      0.88     33000
weighted avg       0.88      0.88      0.88     33000


Wall time: 6min 56s
```



In [22]:
```
%%time
clf_rbf = svm.SVC(kernel='rbf', C=100, gamma=0.001)
clf_rbf.fit(matrix_train, sentiment_train)
print_report(clf_rbf, False)
```

```
Classification report for classifier SVC(C=100, gamma=0.001):
              precision    recall  f1-score   support

    NEGATIVE       0.89      0.92      0.90     16589
    POSITIVE       0.91      0.88      0.90     16411

    accuracy                           0.90     33000
   macro avg       0.90      0.90      0.90     33000
weighted avg       0.90      0.90      0.90     33000


Wall time: 24min 32s
```



We see that both models and are a bit oversensitive to negative reviews. But the difference is very low. In addition, we see that the naive bayes method has remained the same in accuracy (88%), while svc has gone up a bit (90%).

Misclassification

I am very curious to know which sentences one model has trouble with, while the other does not. And which sentences do they both have trouble with. To analyze this, I need the indexes that are misclassified.

In [23]:
```
pred_cnb = clf_cnb.predict(matrix_test.toarray())
pred_rbf = clf_rbf.predict(matrix_test)
```

In [24]:
```
misclassified_cnb = np.where(pred_cnb != sentiment_test)
misclassified_rbf = np.where(pred_rbf != sentiment_test)
```

Now that I have the misclassified indexes, I want to remove indexes when they are in the other array.

In [25]:
```
misclassified_only_rbf = list(set(list(misclassified_rbf[0])) - set(list(mis
misclassified_only_cnb = list(set(list(misclassified_cnb[0])) - set(list(mis
```

I am also curious about the indices that both models got wrong.

In [26]:
```
misclassified_both = list(np.intersect1d(misclassified_cnb, misclassified_rb
```

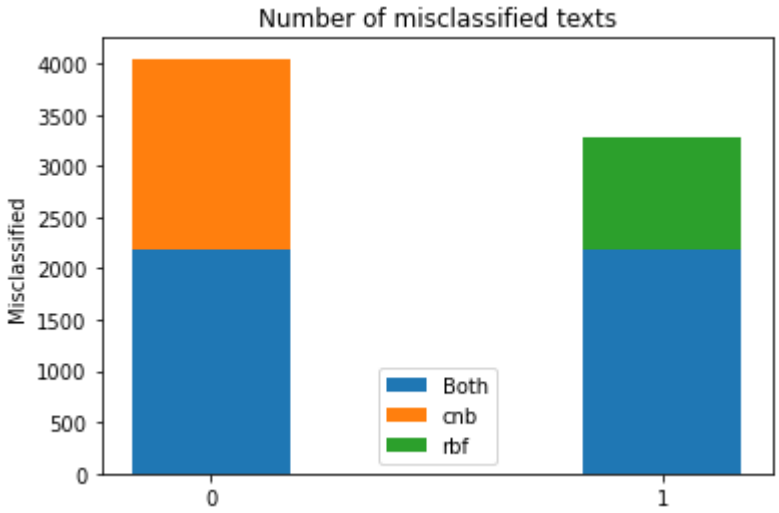
I now have three arrays:

- `misclassified_only_rbf` , these are indexes that only svc rbf got wrong.
- `misclassified_only_cnb` , these are indexes that only Complement naive bayes got wrong.
- `misclassified_both` , these are indexes that both models got wrong.

Distribution in numbers

How is the distribution in numbers?

In [27]:
```
fig, ax = plt.subplots()

width = 0.35

p1 = ax.bar([0,1], [len(misclassified_both),len(misclassified_both)], width,
p3 = ax.bar(0, len(misclassified_only_cnb), width, bottom=len(misclassified_
p2 = ax.bar(1, len(misclassified_only_rbf), width, bottom=len(misclassified_

ax.set_ylabel('Misclassified')
ax.set_title('Number of misclassified texts')
ax.set_xticks([0,1])
ax.legend()


plt.show()
```



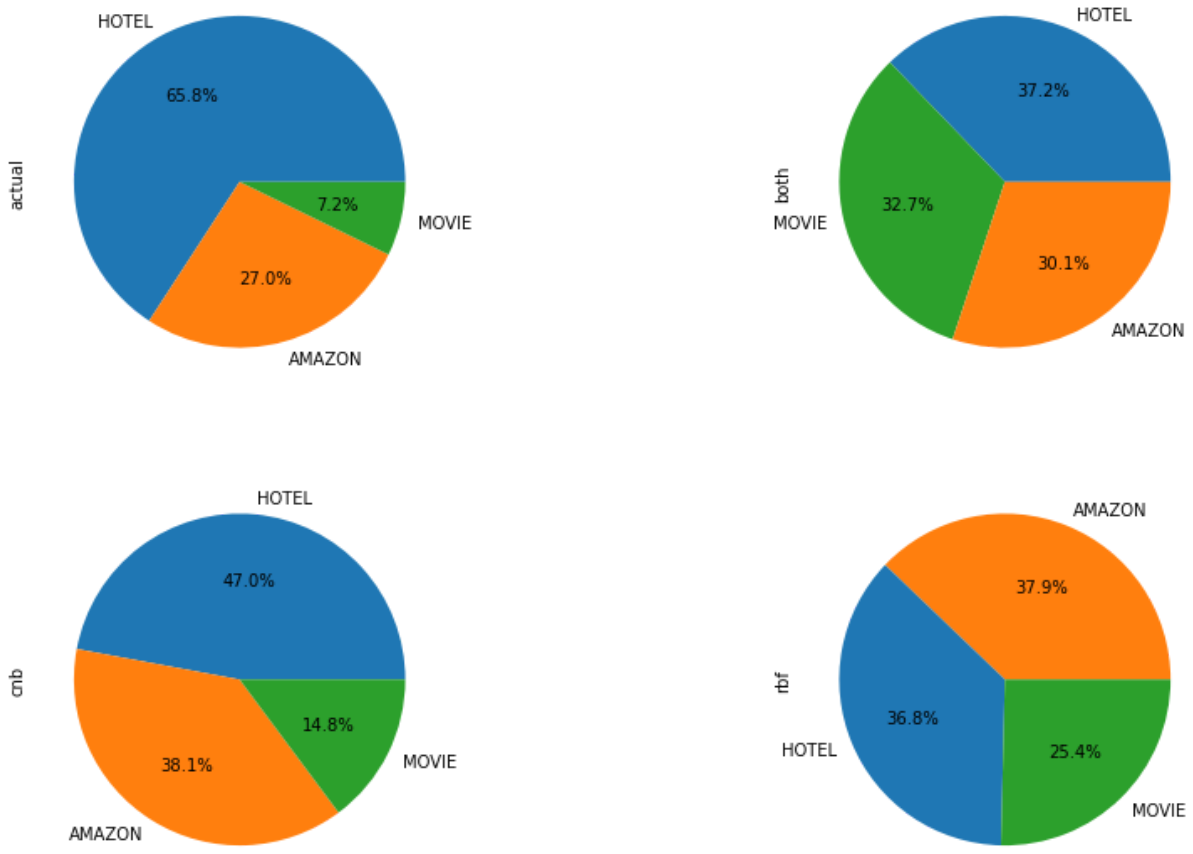
You can see here that both models often struggle with the same sentences. So it may mean that these test sentences are indeed very difficult to classify. And that the models are actually strong.

source

Has the source impact on the model?

In [28]:

```
fig, axs = plt.subplots(2,2)

fig.set_figheight(10)
fig.set_figwidth(15)

colors={'HOTEL':'#1F77B4',
        'AMAZON':'#FF7F0E',
       'MOVIE': '#2CA02C'}


pd.Series(source_test, name='actual').value_counts().plot(ax=axs[0,0], kind=
            colors=[colors[v] for v in pd.Series(source_test).value_counts()
pd.Series(source_test[misclassified_both], name='both').value_counts().plot(
            colors=[colors[v] for v in pd.Series(source_test[misclassified_b
pd.Series(source_test[misclassified_only_cnb], name='cnb').value_counts().pl
            colors=[colors[v] for v in pd.Series(source_test[misclassified_o
pd.Series(source_test[misclassified_only_rbf], name='rbf').value_counts().pl
            colors=[colors[v] for v in pd.Series(source_test[misclassified_o
plt.show()
```



You see four diagrams here. `actual` is the distribution of the source as it really is. `both` is the distribution of the source of all misclassified by both models. `rbf` and `cnb` are the distributions of the source by model.

If a cake piece is larger than with actual, it means that a model has more difficulties with this than with other sources. So both models have a lot of trouble with the movie source.

So the source actually has an impact on the result. Probably the results would have been better if we had left out the movie dataset.

Conclusion

We tried many different options and actually found 2 almost equal performing models. Since rbf SVM has a slightly higher accuracy score, our plan is to use this model for the deployment phase.

Save objects

Since we need to convert custom sentences to a matrix, we need the vectorizer. Additionally, we also need the clasifier. We're going to save both of these objects so we can use them later.

In [29]:
```
with open('obj/vect.pkl', 'wb') as f:
    pickle.dump(vect, f)
with open('obj/clf.pkl', 'wb') as f:
    pickle.dump(clf_rbf, f)
```

Deployment

Project

Several sites allow you to post a comment or opinion about a product, place, or event. Think of Facebook, Instagram, Youtube, or Reddit. However, on not all of these sites, you can see at a glance how many of these comments are written with a positive or negative mindset.

For many people or companies, it is useful to know what percentage of the reactions are positive or negative. You can't always tell this by looking at a Like system. The main question you should be able to answer with the help of this project is; Is a piece of text positive or negative?

Document goal

In this paper, we are going to use the previously created classifier and count vectorizer to complete the deployment phase. This will require the following steps:

1. Import objects.
2. Create a function that classifies a custom text
3. Deploy to web.

Import object

To import objects we use pickle.

In [1]:
```
import pickle
```

In [2]:
```
with open('obj/vect.pkl', 'rb') as f:
    vect = pickle.load(f)
with open('obj/clf.pkl', 'rb') as f:
    clf = pickle.load(f)
```

Create function

In [3]:
```
def classify_custom_text(review):
    #convert text to matrix
    matrix = vect.transform([review])

    #use matrix to make sentiment prediction
    result = clf.predict(matrix.toarray())

    #return result
    return "sentence: " + review + '\nprediction: ' + result[0] +'\n'

```

Test function

In [9]:
```
sentences= [
    "I do not like this at all!",
    "this was the best day ever",
    "I do not agree with this person",
    "places like this make me anxious",
    "This funhouse was the best experience ever"
]

for sentence in sentences:
    print(classify_custom_text(sentence))

```

```
sentence: I do not like this at all!
prediction: NEGATIVE

sentence: this was the best day ever
prediction: POSITIVE

sentence: I do not agree with this person
prediction: NEGATIVE

sentence: places like this make me anxious
prediction: NEGATIVE

sentence: This funhouse was the best experience ever
prediction: POSITIVE
```

The function seems to work great!

Deploy

To make the function public to everyone we use anvil. Anvil is a drag and drop UI creation tool. Any button or text field you can link to a python function.

With anvil you can also make a websocket connection to another (in this case local) server. For this you need a connection string, also called an uplink.

step 1: Read uplink from file

In [4]:
```
anvil_uplink = open("anvil-uplink.txt", "r").read()
```

step 2: Connect via websockets to remote anvil server.

In [5]:
```
import anvil.server
anvil.server.connect(anvil_uplink)
```

```
Connecting to wss://anvil.works/uplink
Anvil websocket open
Connected to "Default environment (dev)" as SERVER
```

step 3: Create callable to use in on anvil side

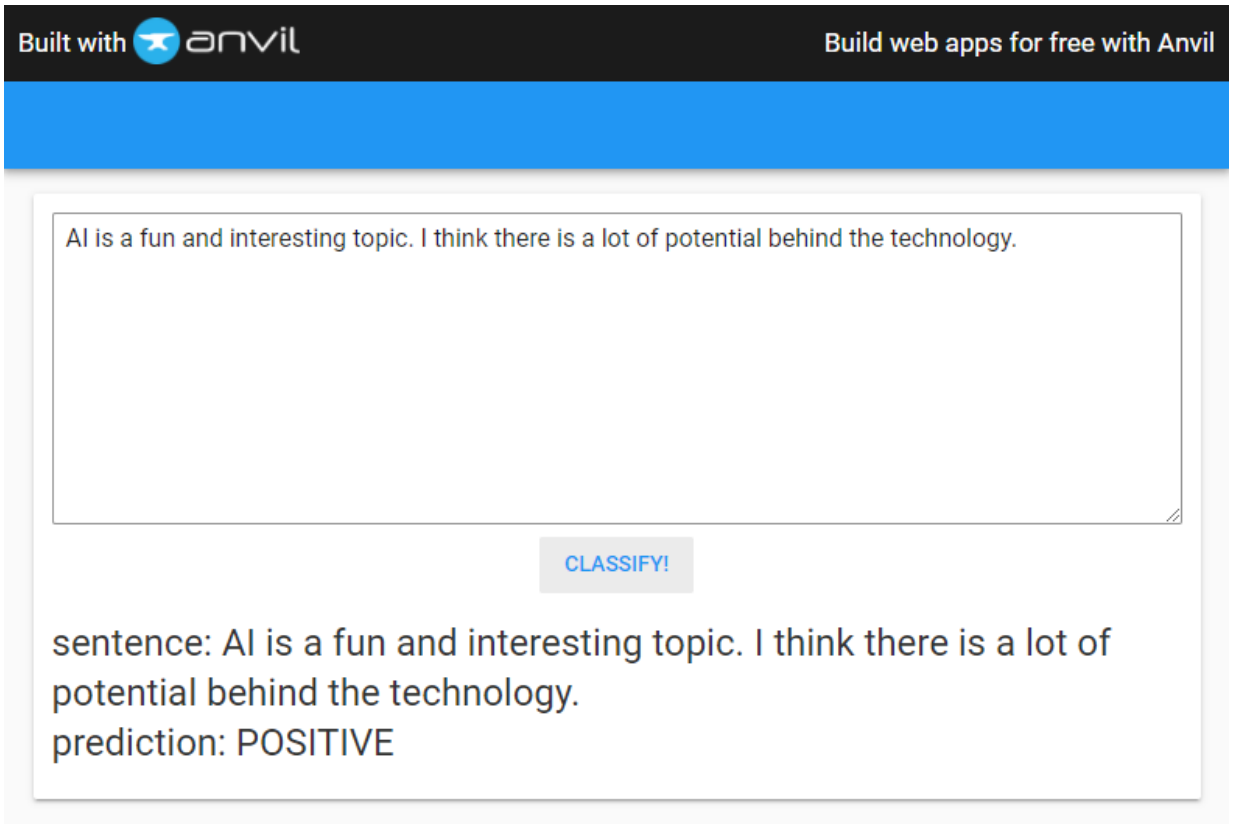
In [6]:
```
@anvil.server.callable
def classify_text(text):
    return classify_custom_text(text)
```

step 4: Use callable on anvil side (screenshot from anvil)

```
from ._anvil_designer import Classifier_FormTemplate
from anvil import *
import anvil.server

class Classifier_Form(Classifier_FormTemplate):
  def __init__(self, **properties):
    self.init_components(**properties)

  def classify_text(self, **event_args):
    result = anvil.server.call('classify_text', self.Text_input.text)
    self.result_lbl.text = result
    pass

```

Conclusion

We have converted the model into a working function and the function can be used via a UI (as long as this notebook is running somewhere). Now we can use go to [this (https://qpdm4eyb7yuprave.anvil.app/7HNS5VZHULNXXQM7BRX5RC63)](https://qpdm4eyb7yuprave.anvil.app/7HNS5VZHULNXXQM7BRX5RC63) website and test our model



Sentiment classifier

Text sentiment, if only there was a way to see this quickly? Is a piece of text positive or negative? The goal is to have the ability to answer this question quickly.

Reviews

Reviews have two unique features that are a perfect fit for this project. They consist of **text**, which can be labeled as positive or negative. At the same time, they also have a **5-star rating** that can be translated into a positive or negative label.

Positive

- Present tense
- Addidtions
- More words

Negative

- Past tense
- Contradictions

90% Accurate

The model can classify with a precision score of 90% whether text is positive or negative.

Try it for yourself now!

Would you like to try it for yourself? Then you can try it here!

Luuk de Kinderen