DOCUMENT SUMMARIZATION (ABSTRACTIVE VS EXTRACTIVE)

Yehia ELnwehy

Md Mahfuz Hasan Shohag

Anirudh Singal

Model statement

The focus is to summarize an input document using an extractive or abstractive method.

The model also extracts highlighted portions of the document (if they exist).

The output is a summarized document + the highlighted portions (Main points defined by user).

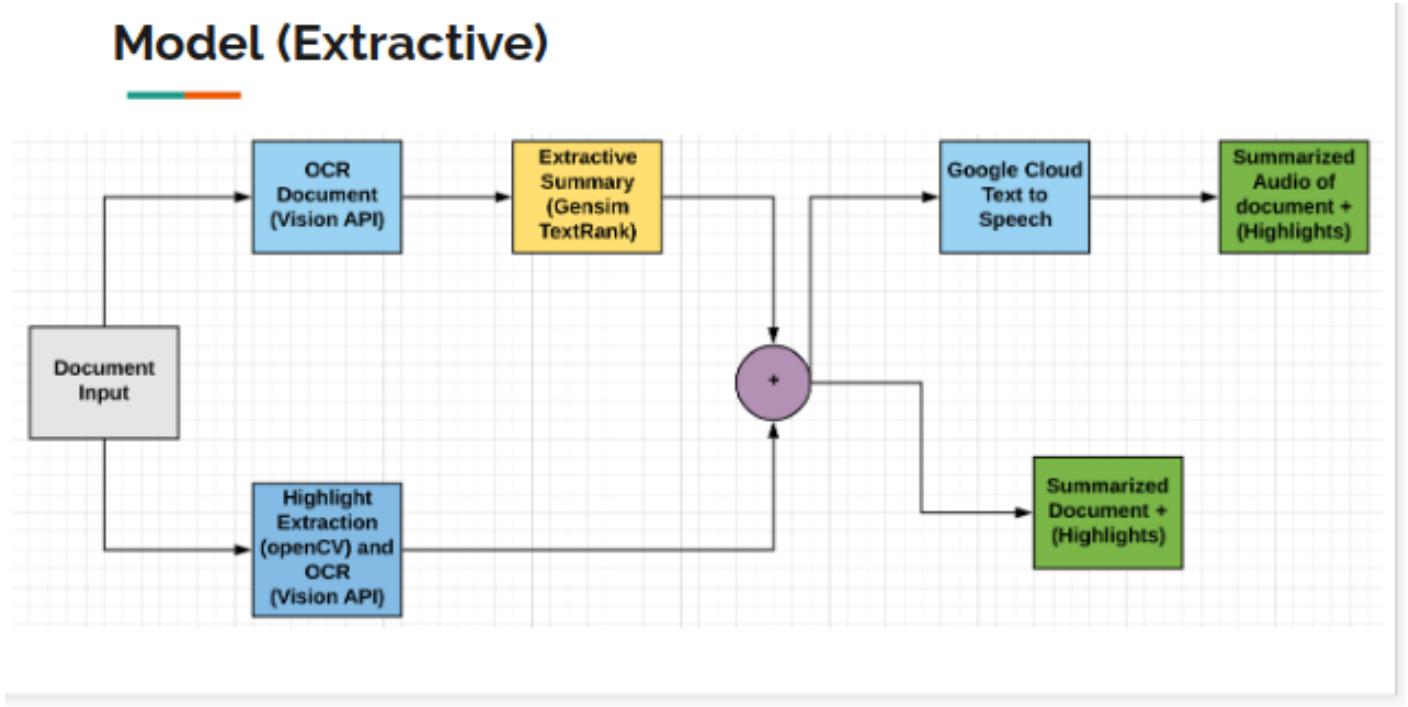
Applications can include generating summaries for news stories or ebooks.

EXTRACTIVE SUMMARIZATION

In this summarization task, the automatic system extracts objects from the entire collection, without modifying the objects themselves. Examples of this include keyphrase extraction, where the goal is to select individual words or phrases to "tag" a document, and document summarization, where the goal is to select whole sentences (without modifying them) to create a short paragraph summary. Similarly, in image collection summarization, the system extracts images from the collection without modifying the images themselves.

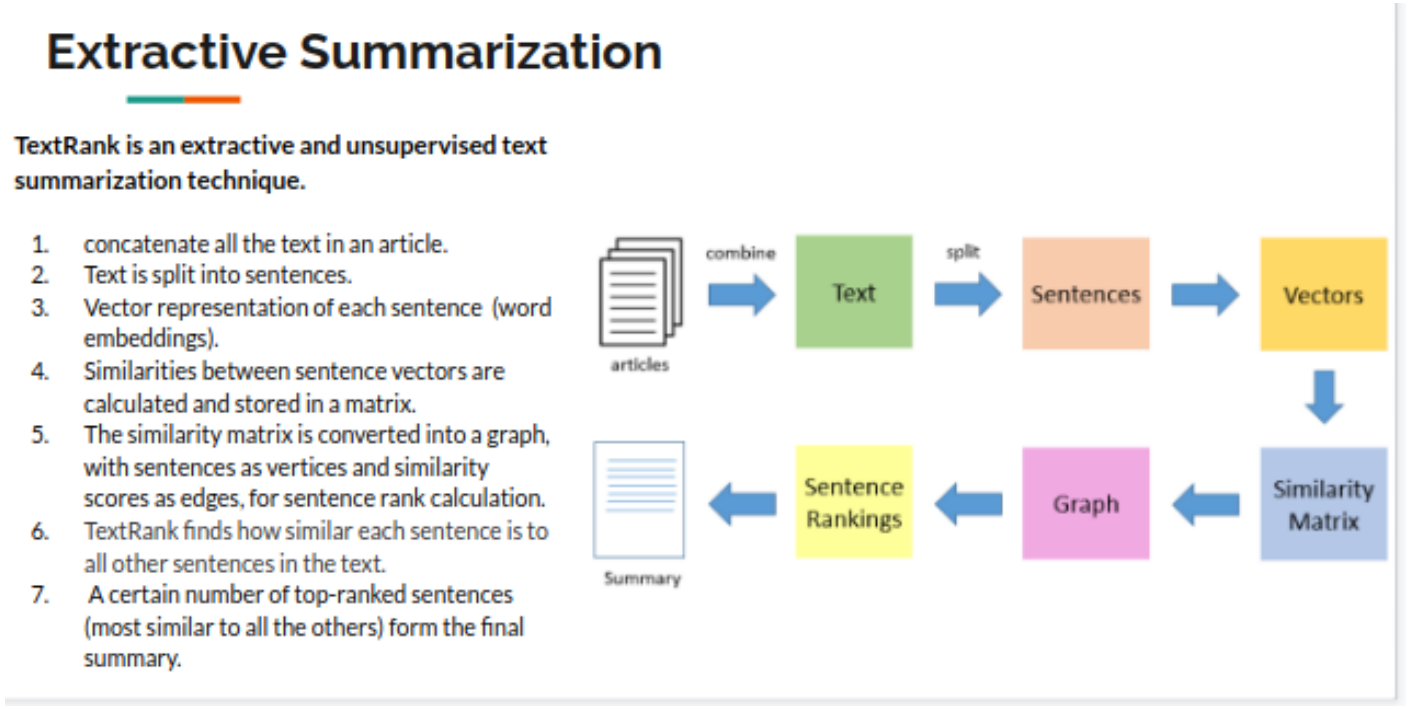
In [14]:
```
from IPython.display import Image
Image(filename='/home/mahfuz/Desktop/2229.png', width=700, height =900)
```

Out[14]:



In [8]:
```
from IPython.display import Image
Image(filename='/home/mahfuz/Desktop/2231.png', width=700, height =900)
```

Out[8]:



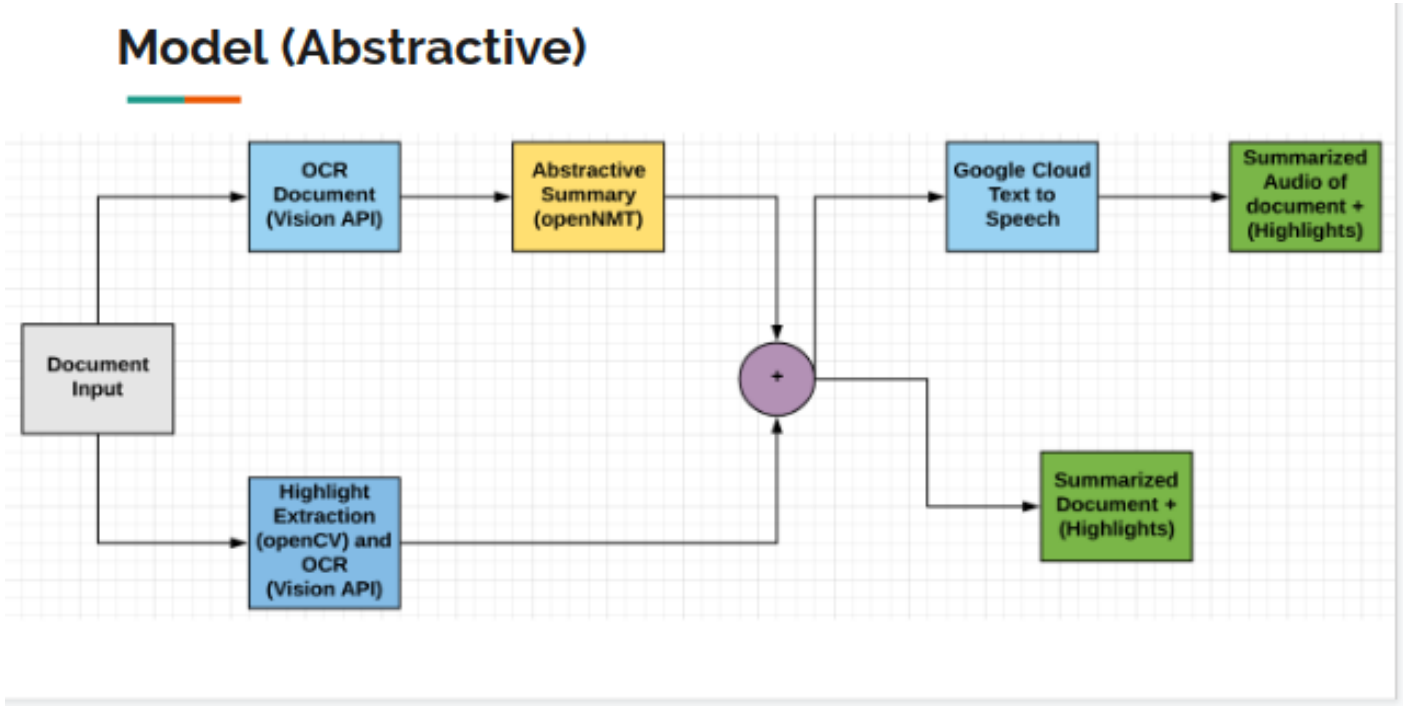
ABSTRACTIVE SUMMARIZATION

Extraction techniques merely copy the information deemed most important by the system to the summary (for example, key clauses, sentences or paragraphs), while abstraction involves paraphrasing sections of the source document. In general, abstraction can condense a text more strongly than extraction, but the programs that can do this are harder to develop as they require use of natural language generation technology, which itself is a growing field.

While some work has been done in abstractive summarization (creating an abstract synopsis like that of a human), the majority of summarization systems are extractive (selecting a subset of sentences to place in a summary).

In [7]:
```
from IPython.display import Image
Image(filename='/home/mahfuz/Desktop/2230.png', width=700, height =900)
```

Out[7]:



In [ ]:
```
#import numpy, openCV,pylab, matplotlib
import numpy as np
import cv2 as cv
import pylab
from matplotlib import pyplot as plt
from PIL import Image

#import gensim for extractive text summarization
import gensim
import matplotlib as mpl

#image resolution to 300
mpl.rcParams['figure.dpi']=300
from google.colab import files

#indtsll cloud vision and text to speech
!pip install --upgrade google-cloud-vision
!pip install --upgrade google-cloud-texttospeech

#install requiremnts for PyTorch
!pip install folium==0.2.1
!pip install imgaug==0.2.6
!pip install numpy==1.15.0
!pip install pandas==0.23
```

In [35]:
```
# input image
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving 4.jpg to 4.jpg
```

In [11]:
```
# google vision api json
uploaded = files.upload()
```

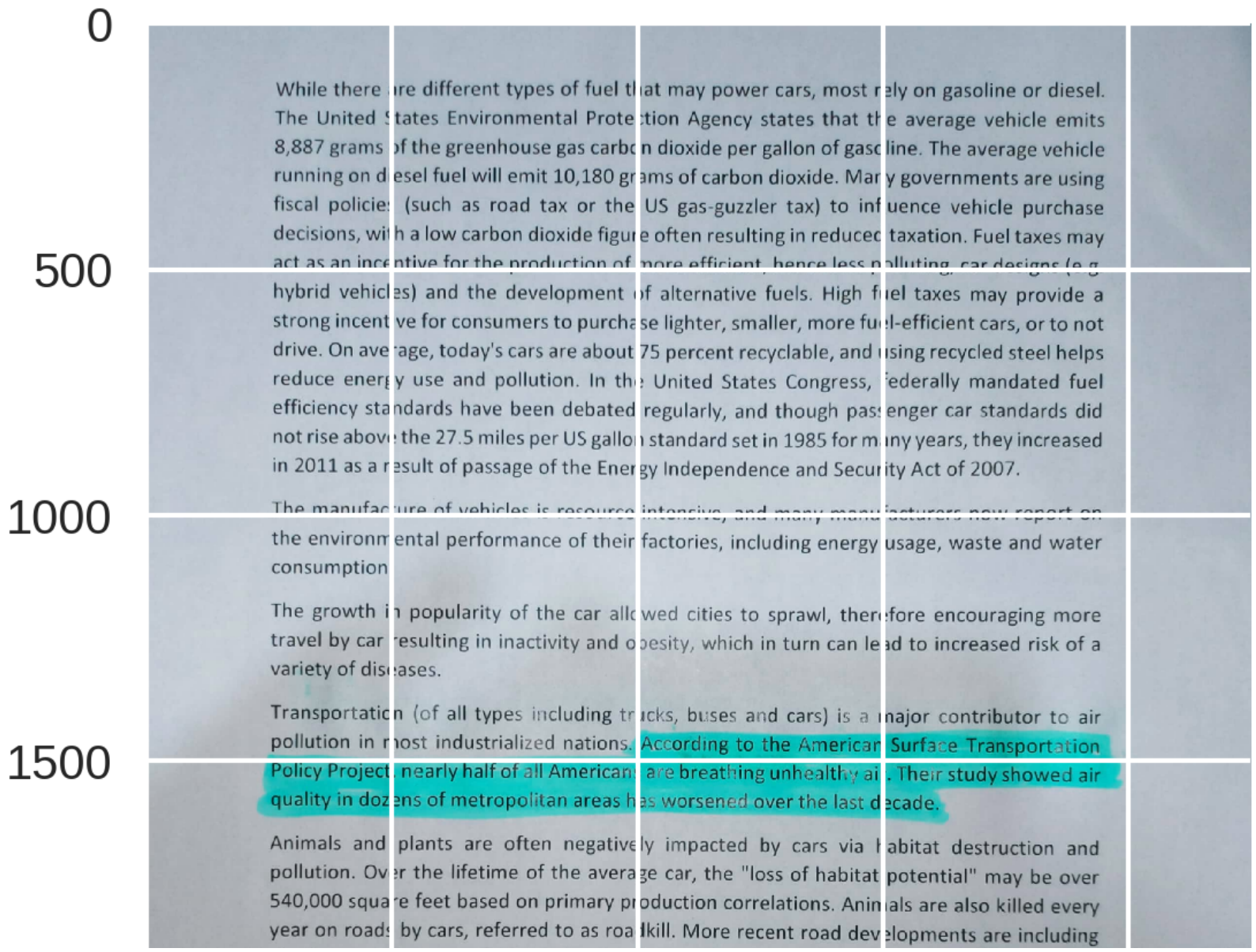
Choose Files No file chosen

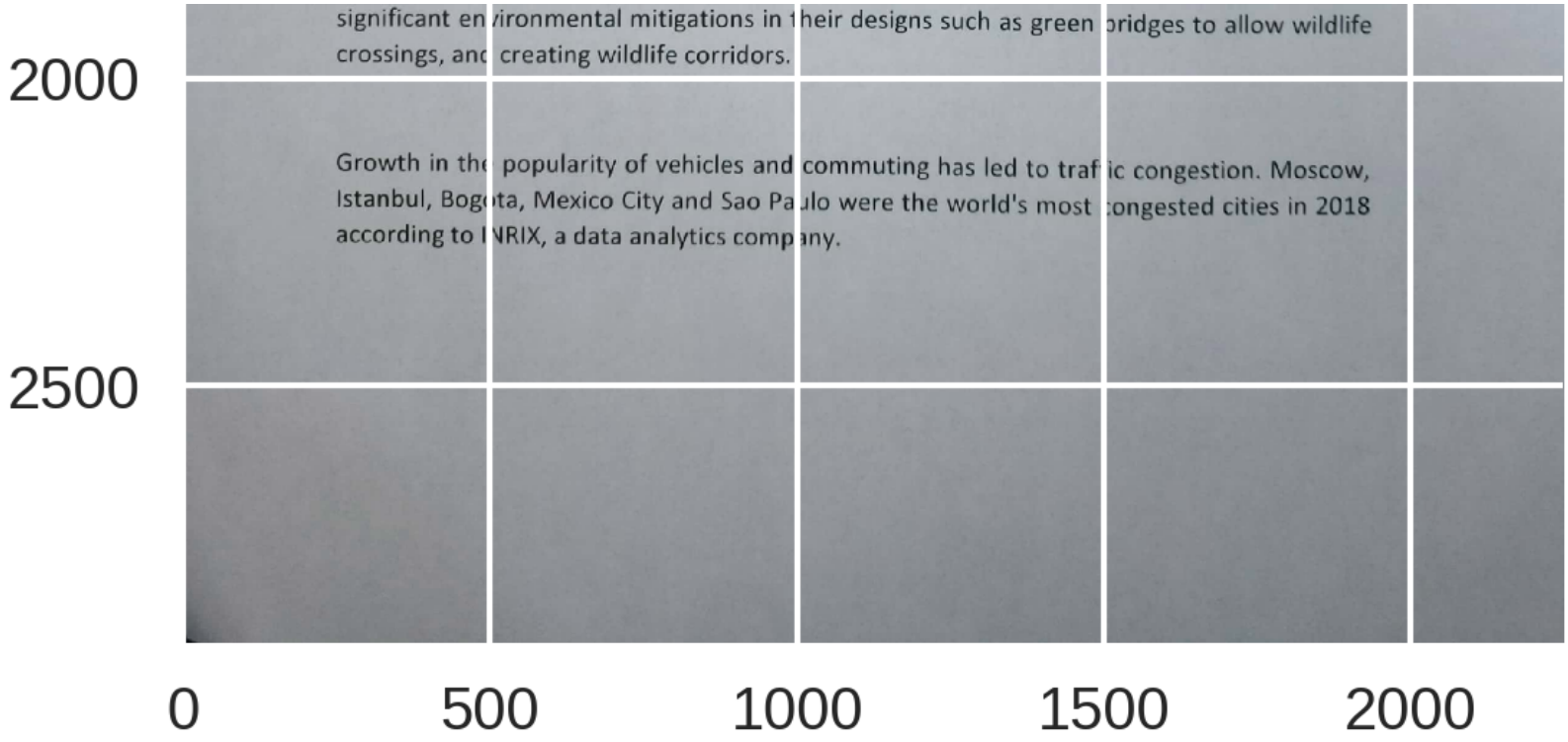
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving My First Project-cecda5b10263.json to My First Project-cecda5b10263 (2).json
```

In [36]:
```
#image path and reading image
img_path = '4.jpg'
img= cv.imread(img_path)
plt.imshow(img)
```

Out[36]: <matplotlib.figure.AxesImage at 0x7f65994497f0>





Highlight Extraction (openCV)

Highlight extraction is done to segregate highlighted text in the document.

“inRange” function in openCV is used to find the boundaries of highlighted portions in the image.

All regions outside this boundary is converted into whitespace.

In [37]:

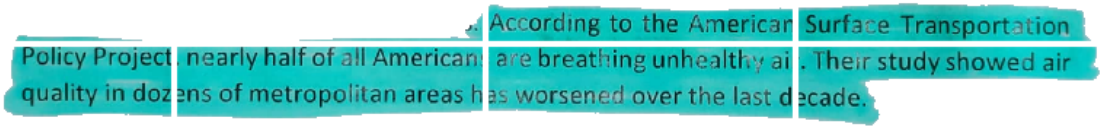
```
#rgb to HSV color spave conversion
hsv_img = cv.cvtColor(img, cv.COLOR_BGR2HSV)

#highlight colour range
hsv_lower=[22, 60, 30]
hsv_upper=[45, 255, 255]

#upper and lower
HSV_lower = np.array(hsv_lower, np.uint8)  # Lower HSV value
HSV_upper = np.array(hsv_upper, np.uint8)  # Upper HSV value

#Threshold
mask_inv = cv.inRange(hsv_img, HSV_lower, HSV_upper)

# find connected components
_,contours, hierarchy =  cv.findContours(mask_inv, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)
contours = [x for x in contours if cv.contourArea(x) > 1000.0]

stencil = np.zeros(img.shape).astype(img.dtype)
cv.fillPoly(stencil, contours, [255,255,255])
stencil_inv = cv.bitwise_not(stencil)
img = cv.bitwise_or(img, stencil_inv)

#cv.drawContours(img, xcontours, -1, (0, 255, 0), 3)
cv.imwrite('tst.jpg',img)
plt.imshow(img)
#print(xcontours)
```

Out[37]: <matplotlib.image.AxesImage at 0x7f6599c5eb00>

500

1000

1500



2000

2500

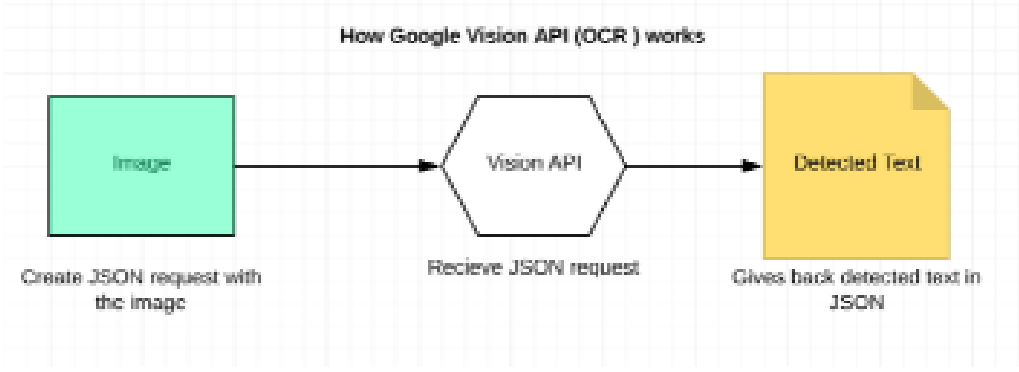
0 500 1000 1500 2000

In [5]:
```
from IPython.display import Image
Image(filename='/home/mahfuz/Desktop/2228.png', width=700, height =900)
```

Out[5]:

Google Vision API

- The Google Cloud Vision API takes complex machine learning models centered around image recognition and formats it in a simple REST API interface.
- It uses a model trained on a large dataset of images, similar to the models used to power Google.
- An API takes a request and tells the system what we want it to do then brings the response back.

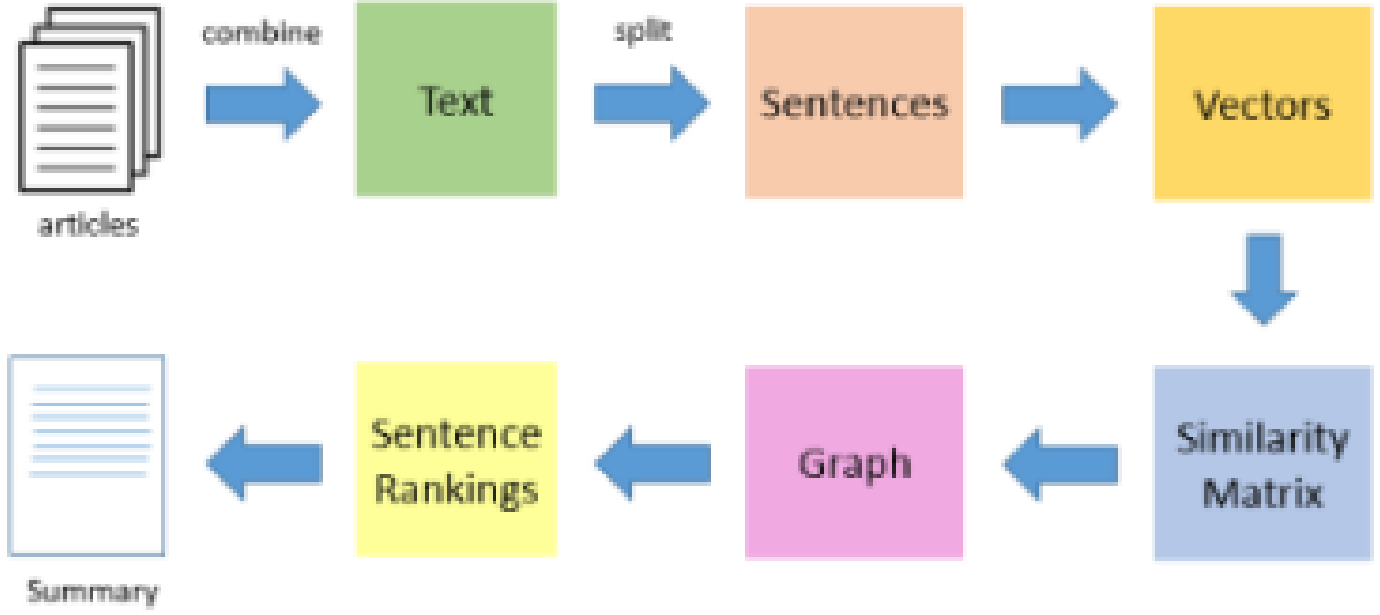


In [0]:
```
def detect_text(path):

    from google.cloud import vision
    client = vision.ImageAnnotatorClient()
    #open image
    import io
    with io.open(path, 'rb') as image_file:
        content = image_file.read()
    #pass image to google vision api
    image = vision.types.Image(content = content)

    #create response type
    response = client.text_detection(image = image)
    texts = response.text_annotations
    #print(texts)
    return texts[0].description.rstrip()
```

Extractive text summarization TextRank is an extractive and unsupervised text summarization technique. 1.concatenate all the text in an article. 2.Text is split into sentences. 3.Vector representation of each sentence (word embeddings). 4.Similarities between sentence vectors are calculated and stored in a matrix. 5.The similarity matrix is converted into a graph, with sentences as vertices and similarity scores as edges, for sentence rank calculation. 6.TextRank finds how similar each sentence is to all other sentences in the text. 7.A certain number of top-ranked sentences (most similar to all the others) form the final summary.

In [10]:

Out[10]:



In [0]:
```
#defining a function that uses gensim for extractive text summarization
def summarize(txt):
    return gensim.summarization.summarize(txt, ratio=0.3)
```

In [0]:
```
#function to connect to google vision API
def connect_to_google_vision_api(path):
    from google.cloud import vision
    import os
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"]= path
```

In [38]:
```
path = 'tst.jpg'
connect_to_google_vision_api("My First Project-cecda5b10263.json")

print('Page Summary:')
print(summarize(detect_text(img_path)))

print('\nHighlighted Points:')
print(detect_text(path))
```

```
Page Summary:
While there are different types of fuel that may power cars, most rely on gasoline or diesel.
The United States Environmental Protection Agency states that the average vehicle emits
running on diesel fuel will emit 10,180 grams of carbon dioxide.
fiscal policies (such as road tax or the US gas-guzzler tax) to influence vehicle purchase
act as an incentive for the production of more efficient, hence less polluting, car designs (e.g
hybrid vehicles) and the development of alternative fuels.
strong incentive for consumers to purchase lighter, smaller, more fuel-efficient cars, or to not
The growth in popularity of the car allowed cities to sprawl, therefore encouraging more
Transportation (of all types including trucks, buses and cars) is a major contributor to air
Animals and plants are often negatively impacted by cars via habitat destruction and
Over the lifetime of the average car, the "loss of habitat potential" may be over
year on roads by cars, referred to as roadkill.
Growth in the popularity of vehicles and commuting has led to traffic congestion.

Highlighted Points:
olicy Project, nearly half of all Americans are breathing unhealthy air. The
According to the American Surface Transportation
ir study showed air
P
quality in dozens of metropolitan areas has worsened over the last decade.
```

In [0]:
```
#page summary uses the extractive summarization function
page_summary = "Page Summary "+summarize(detect_text(img_path))
```

Google Cloud Text-to-Speech converts text into human-like speech in more than 100 voices across 20+ languages and variants. It applies groundbreaking research in speech synthesis (WaveNet) and Google's powerful neural networks to deliver high-fidelity audio.

In [20]:
```
#texttospeech

from google.cloud import texttospeech

# Instantiates a client
client = texttospeech.TextToSpeechClient()

# Set the text input to be synthesized
synthesis_input = texttospeech.types.SynthesisInput(text=page_summary)

# Build the voice request, select the language code ("en-US") and the ssml
# voice gender ("neutral")
voice = texttospeech.types.VoiceSelectionParams(
    language_code='EN',
    ssml_gender=texttospeech.enums.SsmlVoiceGender.NEUTRAL)

# Select the type of audio file you want returned
audio_config = texttospeech.types.AudioConfig(
    audio_encoding=texttospeech.enums.AudioEncoding.MP3)

# Perform the text-to-speech request on the text input with the selected
# voice parameters and audio file type
response = client.synthesize_speech(synthesis_input, voice, audio_config)


# The response's audio_content is binary.
with open('pagesummary.mp3', 'wb') as out:
    # Write the response to the output file.
    out.write(response.audio_content)
    print('Audio content written to file "pagesummary.mp3"')
```

```
Audio content written to file "pagesummary.mp3"
```

In [0]:
```
#implementing vision api OCR
Highlighted_text='Highlighted Points:'+(detect_text(path))
```

In [22]:
```
from google.cloud import texttospeech

# Instantiates a client
client = texttospeech.TextToSpeechClient()

# Set the text input to be synthesized
synthesis_input = texttospeech.types.SynthesisInput(text=Highlighted_text)

# Build the voice request, select the language code ("en-US") and the ssml
# voice gender ("neutral")
voice = texttospeech.types.VoiceSelectionParams(
    language_code='EN',
    ssml_gender=texttospeech.enums.SsmlVoiceGender.NEUTRAL)

# Select the type of audio file you want returned
audio_config = texttospeech.types.AudioConfig(
    audio_encoding=texttospeech.enums.AudioEncoding.MP3)

# Perform the text-to-speech request on the text input with the selected
# voice parameters and audio file type
response = client.synthesize_speech(synthesis_input, voice, audio_config)


# The response's audio_content is binary.
with open('Highlighted_text.mp3', 'wb') as out:
    # Write the response to the output file.
    out.write(response.audio_content)
    print('Audio content written to file "Highlighted_text.mp3"')
```

```
Audio content written to file "Highlighted_text.mp3"
```

In [24]:
```
#downloading pytorch on google colab
from os import path
from wheel.pep425tags import get_abbr_impl, get_impl_ver, get_abi_tag
platform= '{}{}-{}'.format(get_abbr_impl(),get_impl_ver(),get_abi_tag())
accelerator= 'cu80' if path.exists ('/opt/bin/nvidia-smi') else 'cpu'
!pip install -q http://download.pytorch.org/whl/{accelerator}/torch-1.0.1-{platform}-linux_x86_64.wl
```

```
    100% |████████████████████████████████| 530.8MB 32.7MB/s
```

In [25]:
```
#Cloning openNMT from github
!git clone https://github.com/OpenNMT/OpenNMT-py
!cd OpenNMT-py
```

```
Cloning into 'OpenNMT-py'...
remote: Enumerating objects: 13637, done.
remote: Total 13637 (delta 0), reused 0 (delta 0), pack-reused 13637
Receiving objects: 100% (13637/13637), 145.52 MiB | 28.58 MiB/s, done.
Resolving deltas: 100% (9724/9724), done.
```

In [26]:
```
#install openNMT Pytorch requirements file
!pip install -r /content/OpenNMT-py/requirements.txt
```

```
Collecting git+https://github.com/pytorch/text.git@master#wheel=torchtext (from -r /content/OpenNM
T-py/requirements.txt (line 4))
  Cloning https://github.com/pytorch/text.git (https://github.com/pytorch/text.git) (to revision m
aster) to /tmp/pip-req-build-h39bje7y
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from -r /content/Ope
nNMT-py/requirements.txt (line 1)) (1.11.0)
Collecting tqdm==4.30.* (from -r /content/OpenNMT-py/requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/76/4c/103a4d3415dafc1ddfe6a6624333971756e2d3
dd8c6dc0f520152855f040/tqdm-4.30.0-py2.py3-none-any.whl (https://files.pythonhosted.org/packages/7
6/4c/103a4d3415dafc1ddfe6a6624333971756e2d3dd8c6dc0f520152855f040/tqdm-4.30.0-py2.py3-none-any.wh
l) (47kB)
    100% |████████████████████████████████| 51kB 2.5MB/s
Requirement already satisfied: torch>=1.0 in /usr/local/lib/python3.6/dist-packages (from -r /cont
ent/OpenNMT-py/requirements.txt (line 3)) (1.0.1)
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages (from -r /content/
OpenNMT-py/requirements.txt (line 5)) (0.16.0)
Collecting configargparse (from -r /content/OpenNMT-py/requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/55/ea/f0ade52790bcd687127a302b26c1663bf2e0f2
3210d5281dbfcd1dfcda28/ConfigArgParse-0.14.0.tar.gz (https://files.pythonhosted.org/packages/55/e
a/f0ade52790bcd687127a302b26c1663bf2e0f23210d5281dbfcd1dfcda28/ConfigArgParse-0.14.0.tar.gz)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from torchtext=
=0.4.0->-r /content/OpenNMT-py/requirements.txt (line 4)) (2.18.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from torchtext==0.
4.0->-r /content/OpenNMT-py/requirements.txt (line 4)) (1.15.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from
 requests->torchtext==0.4.0->-r /content/OpenNMT-py/requirements.txt (line 4)) (2019.3.9)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requ
ests->torchtext==0.4.0->-r /content/OpenNMT-py/requirements.txt (line 4)) (2.6)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (fr
om requests->torchtext==0.4.0->-r /content/OpenNMT-py/requirements.txt (line 4)) (1.22)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (fr
om requests->torchtext==0.4.0->-r /content/OpenNMT-py/requirements.txt (line 4)) (3.0.4)
Building wheels for collected packages: configargparse, torchtext
  Building wheel for configargparse (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/aa/9c/ce/7e904dddb8c7595ffbe3409d24455bc5005852850e
36011bda
  Building wheel for torchtext (setup.py) ... done
  Stored in directory: /tmp/pip-ephem-wheel-cache-qoh6oyfm/wheels/47/f9/8d/a9e397ec2629a3fd3219b2e
bc3ec8b55396fd3cf55963a77a5
Successfully built configargparse torchtext
```

```
Installing collected packages: tqdm, configargparse, torchtext
  Found existing installation: tqdm 4.28.1
    Uninstalling tqdm-4.28.1:
      Successfully uninstalled tqdm-4.28.1
  Found existing installation: torchtext 0.3.1
    Uninstalling torchtext-0.3.1:
      Successfully uninstalled torchtext-0.3.1
Successfully installed configargparse-0.14.0 torchtext-0.4.0 tqdm-4.30.0
```

In [28]:
```
#connecting to google drive, where the pre-trained CNNDN model for abstractive summarization is sav
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.
0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.goog
leapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20http
s%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code (https://accounts.goog
le.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuserconten
t.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.
com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googl
eapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.rea
donly&response_type=code)

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

In [29]:
```
#unzipping the CNN-DM folder
!tar -xzf '/content/gdrive/My Drive/'cnndm.tar.gz
```

```
tar (child): /content/gdrive/My Drive/cnndm.tar.gz: Cannot open: No such file or directory
tar (child): Error is not recoverable: exiting now
tar: Child returned status 2
tar: Error is not recoverable: exiting now
```

In [30]:
```
#remove whitespace from text detected from input document
par1= detect_text(img_path)
par1 = par1.replace('\n', ' ').replace('\r', '')
print(par1)

f= open("par1.txt","w+")
f.write(par1)
f.close()
```

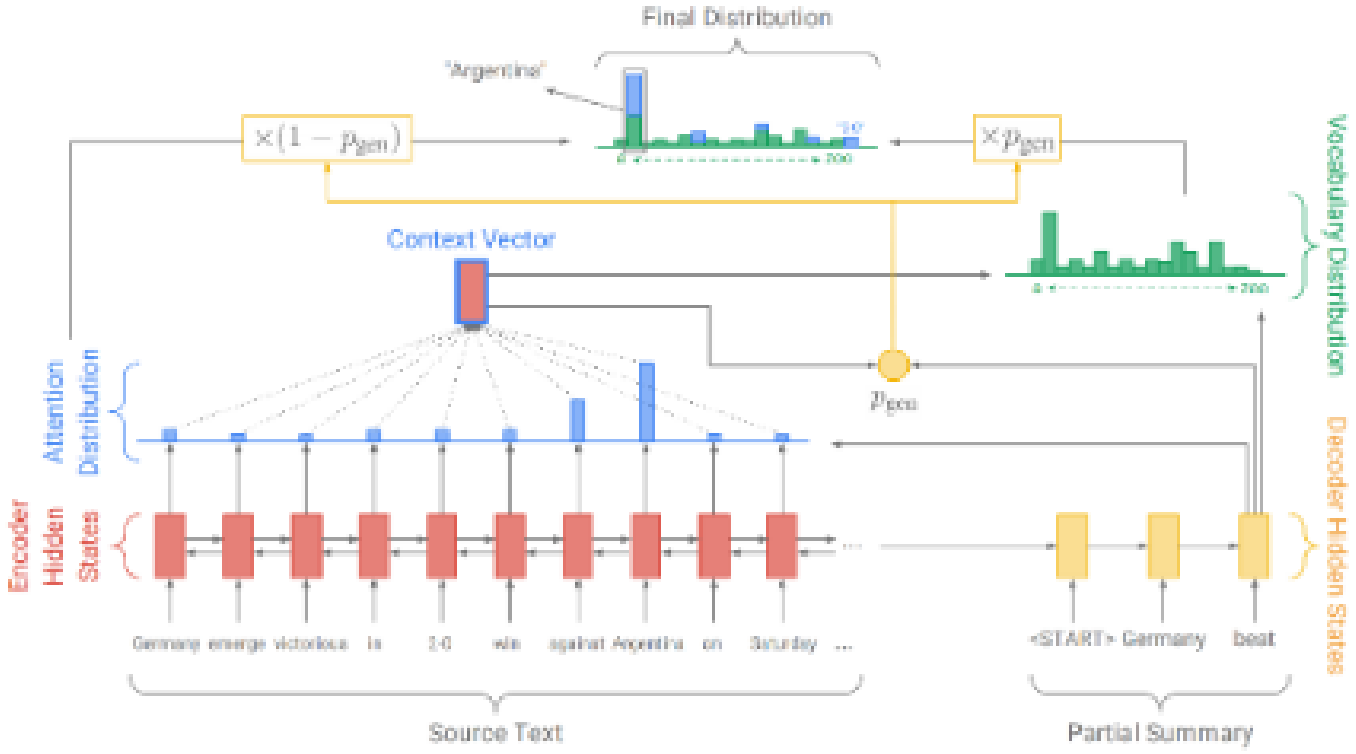
```
A direct impact of widespread adoption of automated vehicles is the loss of driving-related jobs i
n the road transport industry. There could be resistance from professional drivers and unions who
are threatened by job losses. In addition, there could be job losses in public transit services an
d crash repair shops. The automobile insurance industry might suffer as the technology makes certa
in aspects of these occupations obsolete. A frequently cited paper by Michael Osborne and Carl Ben
edikt Frey found that automated cars would make many jobs redundant. Privacy could be an issue whe
n having the vehicle's location and position integrated into an interface in which other people ha
ve access to. In addition, there is the risk of automotive hacking through the sharing of informat
ion through V2V (Vehicle to Vehicle) and V21 (Vehicle to Infrastructure) protocols. There is also
the risk of terrorist attacks. Self-driving cars could potentially be loaded with explosives and u
sed as bombs The lack of stressful driving, more productive time during the trip, and the potentia
l savings in travel time and cost could become an incentive to live far away from cities, where la
nd is cheaper, and work in the city's core, thus increasing travel distances and inducing more urb
an sprawl, more fuel consumption and an increase in the carbon footprint of urban travel. There is
also the risk that traffic congestion might increase, rather than decrease. Appropriate public pol
icies and regulations, such as zoning, pricing, and urban design are required to avoid the negativ
e impacts of increased suburbanization and longer distance travel.
```

Abstractive summarization(Pointer-Generator model):

1.Encoder (Bidirectional LSTM) 2.Decoder 3.Attention distribution 4.Generates from vocabulary distribution or copies from the source based on Pgen (Probability).

In [11]:

Out[11]:

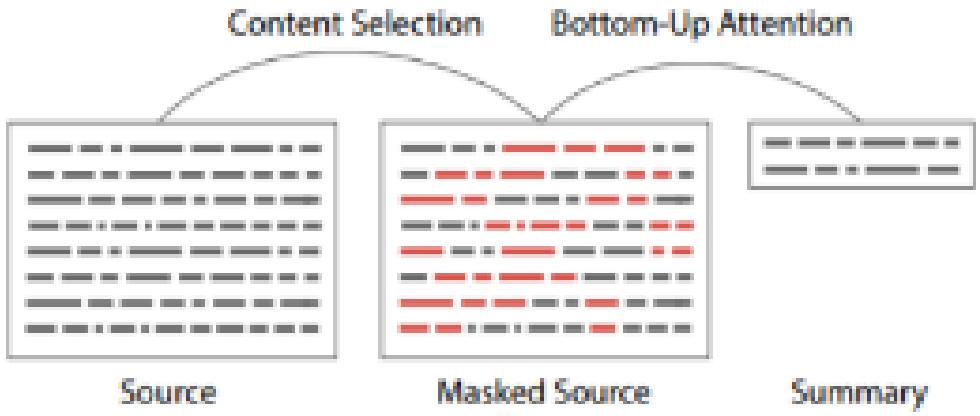


Bottom Up abstractive Summarization

1.Implements an attention mask to limit the available selection of the pointer-generator model. 2.The content selection problem is defined as a word level extractive summarization task.

In [13]:

Out[13]:


Content Selection
Bottom-Up Attention
Source
Masked Source
Summary

In [33]:
```
#abstractive summarization on document using the model
#output to cnndm.out

!python '/content/OpenNMT-py/'translate.py -batch_size 3 \
                    -beam_size 10 \
                    -model '/content/gdrive/My Drive/Abstractive_summarization_yehia/'CNNDM.pt \
                    -src '/content/'par1.txt \
                    -output '/content/'cnndm.out \
                    -min_length 35 \
                    -verbose \
                    -stepwise_penalty \
                    -coverage_penalty summary \
                    -beta 5 \
                    -length_penalty wu \
                    -alpha 0.9 \
                    -verbose \
                    -block_ngram_repeat 3 \
                    -ignore_when_blocking "." "</t>" "<t>"
```

```
[2019-04-06 19:49:53,181 INFO] Translating shard 0.
/usr/local/lib/python3.6/dist-packages/torchtext/data/field.py:359: UserWarning: To copy construct
from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().deta
ch().requires_grad_(True), rather than torch.tensor(sourceTensor).
  var = torch.tensor(arr, dtype=self.dtype, device=device)

SENT 1: ['A', 'direct', 'impact', 'of', 'widespread', 'adoption', 'of', 'automated', 'vehicles',
'is', 'the', 'loss', 'of', 'driving-related', 'jobs', 'in', 'the', 'road', 'transport', 'industr
y.', 'There', 'could', 'be', 'resistance', 'from', 'professional', 'drivers', 'and', 'unions', 'wh
o', 'are', 'threatened', 'by', 'job', 'losses.', 'In', 'addition,', 'there', 'could', 'be', 'job',
'losses', 'in', 'public', 'transit', 'services', 'and', 'crash', 'repair', 'shops.', 'The', 'autom
obile', 'insurance', 'industry', 'might', 'suffer', 'as', 'the', 'technology', 'makes', 'certain',
'aspects', 'of', 'these', 'occupations', 'obsolete.', 'A', 'frequently', 'cited', 'paper', 'by',
'Michael', 'Osborne', 'and', 'Carl', 'Benedikt', 'Frey', 'found', 'that', 'automated', 'cars', 'wo
uld', 'make', 'many', 'jobs', 'redundant.', 'Privacy', 'could', 'be', 'an', 'issue', 'when', 'havi
ng', 'the', "vehicle's", 'location', 'and', 'position', 'integrated', 'into', 'an', 'interface',
'in', 'which', 'other', 'people', 'have', 'access', 'to.', 'In', 'addition,', 'there', 'is', 'th
e', 'risk', 'of', 'automotive', 'hacking', 'through', 'the', 'sharing', 'of', 'information', 'thro
ugh', 'V2V', '(Vehicle', 'to', 'Vehicle)', 'and', 'V21', '(Vehicle', 'to', 'Infrastructure)', 'pro
tocols.', 'There', 'is', 'also', 'the', 'risk', 'of', 'terrorist', 'attacks.', 'Self-driving', 'ca
rs', 'could', 'potentially', 'be', 'loaded', 'with', 'explosives', 'and', 'used', 'as', 'bombs',
'The', 'lack', 'of', 'stressful', 'driving,', 'more', 'productive', 'time', 'during', 'the', 'tri
p,', 'and', 'the', 'potential', 'savings', 'in', 'travel', 'time', 'and', 'cost', 'could', 'becom
e', 'an', 'incentive', 'to', 'live', 'far', 'away', 'from', 'cities,', 'where', 'land', 'is', 'che
```

```
aper,', 'and', 'work', 'in', 'the', "city's", 'core,', 'thus', 'increasing', 'travel', 'distance
s', 'and', 'inducing', 'more', 'urban', 'sprawl,', 'more', 'fuel', 'consumption', 'and', 'an', 'in
crease', 'in', 'the', 'carbon', 'footprint', 'of', 'urban', 'travel.', 'There', 'is', 'also', 'th
e', 'risk', 'that', 'traffic', 'congestion', 'might', 'increase,', 'rather', 'than', 'decrease.',
'Appropriate', 'public', 'policies', 'and', 'regulations,', 'such', 'as', 'zoning,', 'pricing,',
'and', 'urban', 'design', 'are', 'required', 'to', 'avoid', 'the', 'negative', 'impacts', 'of', 'i
ncreased', 'suburbanization', 'and', 'longer', 'distance', 'travel.']
PRED 1: <t> automated cars would make many jobs redundant. Privacy could be an issue when having t
he vehicle's location and position integrated into an interface . </t> <t> traffic congestion migh
t increase, rather than decrease. Appropriate public policies and regulations, such as zoning, pri
cing, and urban design are required to avoid the negative impacts of increased suburbanization .
</t>
PRED SCORE: -2.5873
PRED AVG SCORE: -0.0446, PRED PPL: 1.0456
```

References

1.Gehrmann, Sebastian, et al. “Bottom-Up Abstractive Summarization.” Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, aclweb.org/anthology/D18-1443.

2.See, Abigail, et al. “Get To The Point: Summarization with Pointer-Generator Networks.” Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, doi:10.18653/v1/p17-1099.

3.“Abigail See.” Taming Recurrent Neural Networks for Better Summarization | Abigail See, 2017,

4. www.abigailsee.com/2017/04/16/taming-rnns-for-better-summarization.html (http://www.abigailsee.com/2017/04/16/taming-rnns-for-better-summarization.html).

5.http://opennmt.net/Models-py/ (http://opennmt.net/Models-py/)

6.https://github.com/harvardnlp/sent-summary (https://github.com/harvardnlp/sent-summary)

7.http://opennmt.net/OpenNMT-py/Summarization.html (http://opennmt.net/OpenNMT-py/Summarization.html)

8.https://github.com/OpenNMT/OpenNMT-py (https://github.com/OpenNMT/OpenNMT-py)

9.Joshi, Prateek, and Analytics Vidhya. “Introduction to Text Summarization Using the TextRank Algorithm.”

10.Analytics Vidhya, 13 Dec. 2018, www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/ (http://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/).

11.Mortensen, Ólavur. “Text Summarization with Gensim.” Rare-Technologies.com, 2015, rare-technologies.com/text-summarization-with-gensim/.

12.https://github.com/RaRe-Technologies/gensim (https://github.com/RaRe-Technologies/gensim)

13.https://cloud.google.com/text-to-speech/ (https://cloud.google.com/text-to-speech/)