# TEXT EXTRACTION FROM IMAGES AND SUMMARIZATION WITH ANALYSIS USING NATURAL LANGUAGE PROCESSING

*Submitted in partial fulfillment of the requirements for the degree of*

**Bachelor of Technology**

**in**

**Information Technology**

*by*

**Diti Jain - 21BIT0311**

**Chirag Kukreja - 21BIT0067**

**Under the guidance of**

**Dr. KARTHIKEYAN J**

**SCORE**

**VIT, Vellore.**

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

**September, 2024**

# DECLARATION

**I hereby declare that the thesis entitled "TEXT EXTRACTION FROM IMAGES AND SUMMARIZATION WITH ANALYSIS USING NATURAL LANGUAGE PROCESSING " submitted by me, for the award of the degree of *Bachelor of Technology in Information Technology* VIT is a record of bonafide work carried out by me under the supervision of Dr. KARTHIKEYAN J.**

**I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.**

**Place : Vellore**

**Date : 02/09/2024**

**Signature of the Candidate**

## CERTIFICATE

This is to certify that the thesis entitled "TEXT EXTRACTION FROM IMAGES AND SUMMARIZATION WITH ANALYSIS USING NATURAL LANGUAGE PROCESSING " submitted by Dhruv Jain (21BIT0311) , Chirag Kukreja (21BIT0067) SCORE VIT, for the award of the degree of *Bachelor of Technology in Information Technology*, is a record of bonafide work carried out by him / her under my supervision during the period, 01. 12. 2018 to 30.04.2019, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the university in my opinion meets the necessary standards for submission.

Place: Vellore

Date: 02/09/2024 Signature of the Guide

Internal Examiner External Examiner

# Table of Contents

| Sr. No. | Title | Page no. |
|---|---|---|
| 1 | Literature Review | 5 |
| 2 | Gap Identification | 7 |
| 3 | Objective Framing | 9 |
| 4 | Project Plan | 10 |
| 5 | Implementation and Analysis | 12 |
| 6 | Design and Methodology | 18 |
| 7 | References | 20 |

## 1) Literature Review / Rationale

The field of Natural Language Processing (NLP) encompasses various techniques to handle large volumes of text data, including text extraction, summarization, and classification. This literature review discusses the existing methods and their limitations, providing the rationale for the approaches adopted in this project.

**Text Extraction from Images:**

Traditional Optical Character Recognition (OCR) methods are commonly used for extracting text from images. However, they often fail to handle variations in fonts, sizes, text orientations, and complex backgrounds. Pansare et al. (2019) improved OCR accuracy by integrating preprocessing techniques such as Gabor filters and Support Vector Machines (SVM) to detect text more effectively in diverse types of images . Similarly, Lu et al. (2015) used edge-based extraction methods combined with support vector regression to extract text from noisy and complex backgrounds . These studies highlight the need for enhanced preprocessing methods to improve text recognition accuracy. The current project builds on these advancements by employing FirebaseML for text extraction, which offers greater flexibility in handling various text formats and complex images.

**Text Summarization Techniques:**

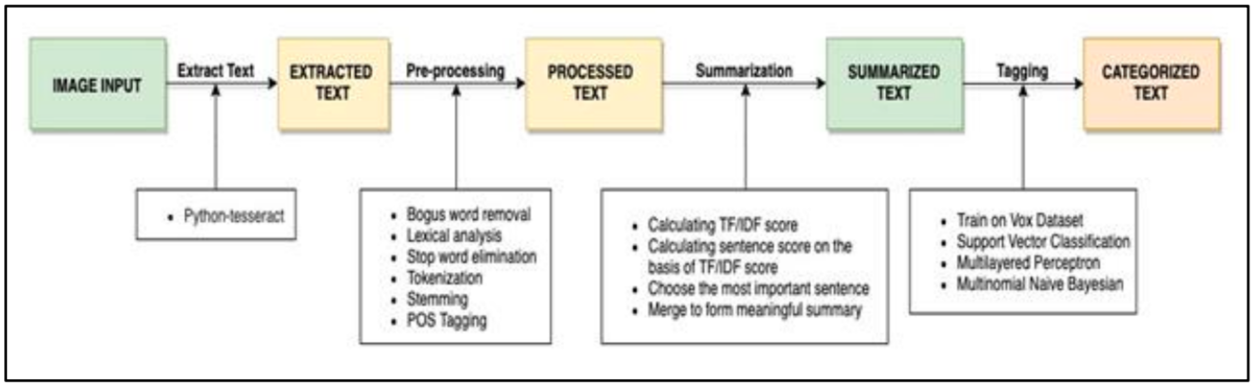
Text summarization can be broadly categorized into extractive and abstractive approaches. Extractive methods select key sentences based on features like word frequency, sentence position, or other statistical measures. Mallick et al. (2019) used algorithms such as PageRank to rank sentences for extractive summarization, which is computationally efficient but may produce summaries that lack coherence. On the other hand, abstractive methods generate new sentences that convey the main ideas of the text. Song et al. (2019) proposed a model combining Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) to achieve more contextually relevant summaries. The current project adopts a hybrid approach, leveraging both extractive methods for efficiency and semantic similarity measures, like cosine similarity, to enhance the quality and coherence of the summaries.

**Text Classification and Tagging:**

Effective classification and tagging of text documents are essential for organizing and filtering information. Traditional classifiers, such as SVM and Naive Bayes, have been widely used for these tasks. Al Qadi et al. (2019) demonstrated high accuracy in classifying news articles using these methods. However, recent advancements like BERT (Bidirectional Encoder Representations from Transformers) have significantly improved text classification by capturing deeper contextual relationships between words. Zhang et al. (2019) demonstrated BERT's superior performance in classification tasks, making it highly suitable for this project's objectives. This project employs BERT for text classification to ensure accurate categorization into relevant domains such as politics, technology, entertainment, and business.

**Rationale for the Chosen Approach:**

The methodologies chosen for this project aim to leverage the strengths of both traditional and modern NLP techniques. By combining OCR with advanced preprocessing methods, using a hybrid approach for summarization, and applying BERT for classification, the project addresses the limitations of each individual method. This comprehensive approach ensures robustness against different types of text data, ranging from structured documents to complex, unstructured images.



## 2) Gap Identification

**Limited Scope of Summarization Techniques:**

- While the document discusses extractive and abstractive summarization, it may not cover recent advancements in hybrid approaches or multi-document summarization. There is a gap in exploring how combining various methods can enhance summarization quality.

**Inadequate Attention to Domain-Specific Summarization:**

- The literature review might lack an examination of domain-specific summarization techniques, which can significantly vary based on the field (e.g., medical, legal, technical). There is an opportunity to explore tailored approaches for different industries.

**Lack of Real-World Applications and Case Studies:**

- The document does not provide examples of real-world applications of the discussed techniques. Including case studies could demonstrate the practical implications and effectiveness of various summarization methods.

**Underexplored Evaluation Metrics:**

- The document may not address the evaluation metrics used to assess the quality of summaries generated. There is a gap in discussing how to effectively measure the performance of summarization systems beyond traditional metrics like ROUGE.

**Challenges in Handling Noisy Data:**

- There is limited discussion on the challenges of summarizing text from noisy or unstructured data sources, such as social media or user-generated content. This presents an opportunity to explore techniques that can effectively handle such data.

**Ethical Considerations and Bias:**

- The implications of bias in NLP models and the ethical considerations surrounding automated summarization are crucial topics that may not be thoroughly examined in the document. Addressing this gap can enhance the understanding of responsible AI use.

**User-Centric Design and Usability:**

- The document may not delve into the user experience aspect of summarization tools. Understanding user needs and preferences in summarization can lead to better-designed systems that cater to specific audiences.

**Integration with Other Technologies:**

- There is potential for exploring how summarization techniques can be integrated with other NLP tasks, such as sentiment analysis or information retrieval, to create more comprehensive solutions.

## 3) Objective Framing

**Efficient Information Processing** :

- To develop a framework that facilitates the extraction and summarization of key information from news articles, helping users quickly grasp important facts.

**Text Extraction and Cleaning** :

- To implement a robust text extraction mechanism that accurately identifies and extracts relevant text from various input image formats while filtering out misleading or erroneous words.

**Text Pre-processing** :

- To enhance the quality and context of the extracted text through pre-processing steps, including lexical analysis, stop word elimination, tokenization, stemming, and POS tagging.

**Summary Generation** :

- To create concise summaries from the pre-processed text that highlight critical points, utilizing techniques such as TF/IDF calculations to determine the significance of sentences.

**Tagging Mechanism** :

- To apply tagging algorithms that categorize the summarized text into relevant domains or topics, assisting users in navigating content according to their interests.

**User-Centric Recommendations** :

- To establish a recommendation system based on user interactions with tagged articles, providing personalized suggestions for related content.

**Time-Saving for Users** :

- To offer a time-efficient solution for users inundated with information, enabling them to quickly access the most pertinent details without needing to read entire articles.

## 4) Project Plan

**1. Project Overview**

- **Project Title:** NLP Framework for Text Extraction and Summarization
- **Objective:** To develop an automated system that extracts key information from news articles, summarizes the content, and categorizes it through tagging.

**2. Project Phases**

- **Phase 1: Research and Requirement Analysis**
- **Duration:** 2 weeks
- **Activities:**
    - Conduct a literature review on existing NLP techniques.
    - Identify user requirements through surveys or interviews with potential users.
    - Define the scope and specifications of the project.
- **Deliverables:** Requirement specification document.
- **Phase 2: Data Collection**
- **Duration:** 3 weeks
- **Activities:**
    - Gather a dataset of news articles from various sources (APIs, web scraping).
    - Ensure the dataset is diverse and covers multiple topics.
- **Deliverables:** Compiled dataset of news articles.
- **Phase 3: Text Extraction and Pre-processing**
- **Duration:** 4 weeks
- **Activities:**
    - Develop and implement text extraction algorithms for different formats (e.g., PDF, HTML).
    - Perform text cleaning and pre-processing (tokenization, stop word removal, stemming).
- **Deliverables:** Cleaned and pre-processed dataset ready for analysis.
- **Phase 4: Summary Generation**
- **Duration:** 4 weeks

- **Activities:**
  - Implement extractive and abstractive summarization techniques.
  - Utilize algorithms such as TF-IDF and neural networks for summary generation.
- **Deliverables:** Summarization module that generates concise article summaries.
- **Phase 5: Tagging and Classification**
- **Duration:** 3 weeks
- **Activities:**
  - Develop a tagging mechanism to categorize articles based on their content.
  - Implement machine learning algorithms for classification (e.g., SVM, decision trees).
- **Deliverables:** Tagging and classification module.
- **Phase 6: User Interface Development**
- **Duration:** 3 weeks
- **Activities:**
  - Design a user-friendly interface for users to input articles and receive summaries and tags.
  - Incorporate features for personalized recommendations based on user preferences.

## 5) Implementation (50%) & Analysis

**1. Image to text**

```
import pytesseract
import cv2
pytesseract.pytesseract.tesseract_cmd = "C:\\Program Files\\Tesseract-OCR\\tesseract"
img = cv2.imread("3.jpg")
scale_percent = 100 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
img = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
text = pytesseract.image_to_string(img)
with open("text.txt","w") as f:
 f.write(text)
print(text)
cv2.imshow("img",img)
cv2.waitKey(0)
# Text Tagging, Summarization and Analysis of articles
```

**2. Text Summarization**

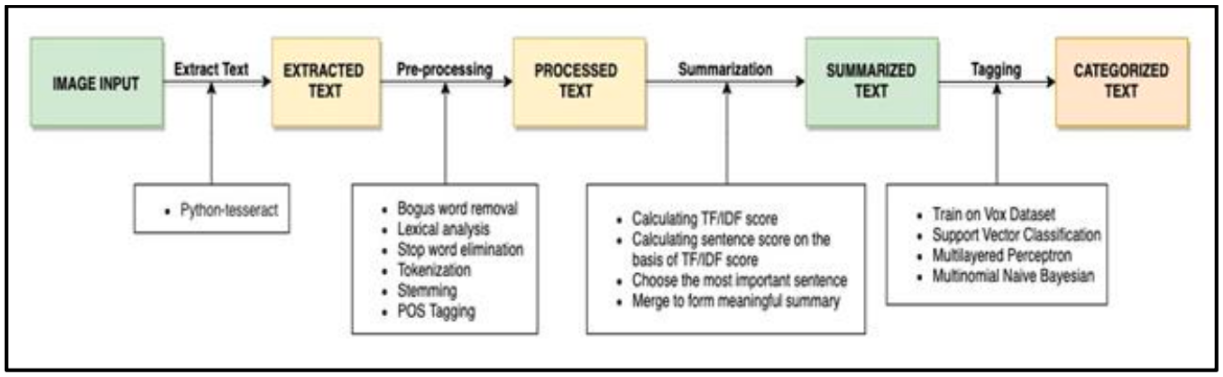
```
import nltk
nltk.download('brown')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
import re
import string
import numpy as np
import matplotlib.pyplot as plt
from nltk import pos_tag
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus.reader.wordnet import NOUN, VERB, ADJ, ADV
from nltk.corpus import brown, stopwords
from nltk.cluster.util import cosine_distance
from operator import itemgetter
"""### Using the corpus text for summarization"""
sentences = brown.sents('ca01')
sentences
len(sentences)
[' '.join(sent) for sent in sentences]
"""# Preprocessing"""
class TextCleaner():

 def __init__(self):
 self.stop_words = set(stopwords.words("english"))
 self.punctuations = set(string.punctuation)
 self.pos_tags = {
 NOUN: ['NN', 'NNS', 'NNP', 'NNPS', 'PRP', 'PRP$', 'WP', 'WP$'],
 VERB: ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ'],
 ADJ: ['JJ', 'JJR', 'JJS'],
 ADV: ['RB', 'RBR', 'RBS', 'WRB']
 }
 def _remove_stop_words(self, words):
 return [w for w in words if w not in self.stop_words]


 def _remove_regex(self):
 self.input_sent = " ".join([w.lower() for w in self.input_sent])
 self.input_sent = re.sub(r"i'm", "i am", self.input_sent)
 self.input_sent = re.sub(r"he's", "he is", self.input_sent)
 self.input_sent = re.sub(r"she's", "she is", self.input_sent)
 self.input_sent = re.sub(r"that's", "that is", self.input_sent)
 self.input_sent = re.sub(r"what's", "what is", self.input_sent)
 self.input_sent = re.sub(r"where's", "where is", self.input_sent)
 self.input_sent = re.sub(r"\'ll", " will", self.input_sent)
 self.input_sent = re.sub(r"\'ve", " have", self.input_sent)
 self.input_sent = re.sub(r"\'re", " are", self.input_sent)
 self.input_sent = re.sub(r"\'d", " would", self.input_sent)
 self.input_sent = re.sub(r"won't", "will not", self.input_sent)
 self.input_sent = re.sub(r"can't", "cannot", self.input_sent)
 self.input_sent = re.sub(r"don't", "do not", self.input_sent)
 patterns = re.finditer("#[\w]*", self.input_sent)
 for pattern in patterns:
 self.input_sent = re.sub(pattern.group().strip(), "", self.input_sent)
 self.input_sent = "".join(ch for ch in self.input_sent if ch not in self.punctuations)


 def _tokenize(self):
 return word_tokenize(self.input_sent)


 def _process_content_for_pos(self, words):
 tagged_words = pos_tag(words)
 pos_words = []
 for word in tagged_words:
 flag = False
 for key, value in self.pos_tags.items():
 if word[1] in value:
 pos_words.append((word[0], key))
 flag = True
 break
 if not flag:
 pos_words.append((word[0], NOUN))
 return pos_words


 def _remove_noise(self):
 self._remove_regex()
```

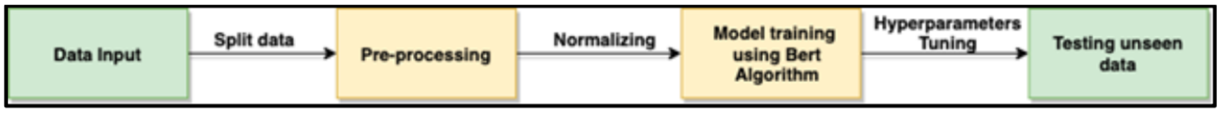
```
 words = self._tokenize()
 noise_free_words = self._remove_stop_words(words)
 return noise_free_words


 def _normalize_text(self, words):
 lem = WordNetLemmatizer()
 pos_words = self._process_content_for_pos(words)
 normalized_words = [lem.lemmatize(w, pos=p) for w, p in pos_words]
 return normalized_words


 def clean_up(self, input_sent):
 self.input_sent = input_sent
 cleaned_words = self._remove_noise()
 cleaned_words = self._normalize_text(cleaned_words)
 return cleaned_words
"""## PageRank Algorithm"""
def pagerank(M, eps=1.0e-8, d=0.85):
 N = M.shape[1]
 v = np.random.rand(N, 1)
 v = v / np.linalg.norm(v, 1)
 last_v = np.ones((N, 1), dtype=np.float32) * np.inf
 M_hat = (d * M) + (((1 - d) / N) * np.ones((N, N), dtype=np.float32))

 while np.linalg.norm(v - last_v, 2) > eps:
 last_v = v
 v = np.matmul(M_hat, v)
 return v
"""## Cosine similarity"""
def sentence_similarity(sent1, sent2):
 text_cleaner = TextCleaner()

 sent1 = text_cleaner.clean_up(sent1)
 sent2 = text_cleaner.clean_up(sent2)

 all_words = list(set(sent1 + sent2))

 vector1 = [0] * len(all_words)
 vector2 = [0] * len(all_words)

 for w in sent1:
 vector1[all_words.index(w)] += 1

 for w in sent2:
 vector2[all_words.index(w)] += 1

 return 1 - cosine_distance(vector1, vector2)
"""## Similarity adjacency matrix"""
def build_similarity_matrix(sentences):
 S = np.zeros((len(sentences), len(sentences)))
 for i in range(len(sentences)):
 for j in range(len(sentences)):
 if i == j:
 continue
 else:
 S[i][j] = sentence_similarity(sentences[i], sentences[j])

 for i in range(len(S)):
 S[i] /= S[i].sum()
 return S
import nltk
nltk.download('punkt')
S = build_similarity_matrix(sentences)
S
sentence_ranks = pagerank(S)
sentence_ranks
ranked_sentence_indexes = [item[0] for item in sorted(enumerate(sentence_ranks), key=lambda item: -
item[1])]
ranked_sentence_indexes
plt.bar([item[0] for item in sorted(enumerate(sentence_ranks))], sentence_ranks.T[0])
plt.xlabel("Sentence No.")
plt.ylabel("Importance")
plt.show()
plt.plot([item[0] for item in sorted(enumerate(sentence_ranks))], sentence_ranks)
plt.xlabel("Sentence No.")
plt.ylabel("Importance")
plt.show()
SUMMARY_SIZE = 5
"""Taking the top 5 sentences for summary"""
selected_sentences = sorted(ranked_sentence_indexes[:SUMMARY_SIZE])
selected_sentences
summary = itemgetter(*selected_sentences)(sentences)
string=""
for sent in summary:
 for s in sent:
 string=string+" "+s
nw=string.split()
print("Num words ",len(nw))
string
```

## 6) Design/Methodology Overview

1. **Input and Output** :

- **Input** : An image of a news article.
- **Output** : Summarization and tagging of the news article.





2. **Modules and Steps** :

### a) Image Input

- Input images may vary in size and are converted to a fixed size to enhance processing efficiency.

### b) Text Extraction from Image

- Text is extracted from the provided images using text extraction tools. Media and non-meaningful text are disregarded.

### c) Pre-processing of Extracted Text

- The extracted text undergoes several processes:
    - Removing erroneous or useless words (bogus words).
    - Performing lexical analysis.
    - Eliminating stop words.
    - Tokenization.
    - Stemming and Part-of-Speech (POS) tagging.

- This pre-processed text ensures improved accuracy for subsequent analysis.

**d) Text Summarization**

- The text is summarized using the TF/IDF (Term Frequency-Inverse Document Frequency) method. Key sentences are identified and merged to create an extractive summary focusing on main points.

**e) Tagging of Summarized Text**

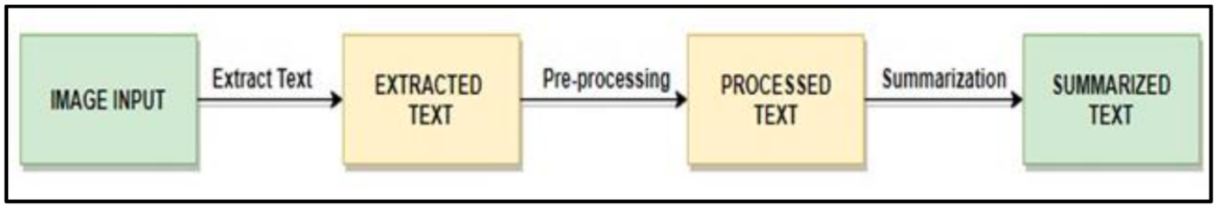
- A tagging algorithm is applied to the summarized text to generate relevant tags, enabling users to filter news articles based on priorities.

**f) Categorization of Text**

- The model employs support vector classification for categorizing news articles based on generated tags. Techniques like multi-layer perceptron and multinomial naive bayes are utilized to group similar tags and categorize articles based on content.

**g) Analysis of the Proposed Model**

- The entire model functions sequentially with each module contributing to the final objective of converting raw news images into summarized text enriched with tags and categorization.



## 7) REFERENCES:

[1] Song, S., Huang, H., & Ruan, T. (2019). Abstractive text summarization using LSTM-CNN based deep learning. Multimedia Tools and Applications, 78(1), 857-875.

[2] Kanapala, A., Pal, S., & Pamula, R. (2019). Text summarization from legal documents: a survey. Artificial Intelligence Review, 51(3), 371-402.

[3] Zhang, H., Gong, Y., Yan, Y., Duan, N., Xu, J., Wang, J., & Zhou, M. (2019). Pretraining-Based Natural Language Generation for Text Summarization. arXiv preprint arXiv:1902.09243.

[4] Pansare, JR., Gaikwad, A., Ankam, V.,Sharma S. & Karne P. (2019) Real-Time Text Reader for English Language. International Research Journal Of Engineering And Technology (IRJET) E-ISSN: 2395-0056 Volume: 06 Issue: 05

[5] Mallick, C., Das, A. K., Dutta, M., Das, A. K., & Sarkar, A. (2019). Graph-Based Text Summarization Using Modified TextRank. In Soft Computing in Data Analytics (pp. 137-146). Springer, Singapore.

[6] Adam, G., Bouras, C., & Poulopoulos, V. (2009, May). CUTER: An efficient useful text extraction mechanism. In *2009*
*International Conference on Advanced Information Networking and Applications Workshops* (pp. 703-708). IEEE.

[7] Futrelle, R. P. (1999). Summarization of diagrams in documents. *Advances in Automated Text Summarization*, 403-421.

[8] Lu, S., Chen, T., Tian, S., Lim, J. H., & Tan, C. L. (2015). Scene text extraction based on edges and support vector regression. *International Journal on Document Analysis and Recognition (IJDAR)*, *18*(2), 125-135. [10] Hasan, Yassin MY, and
Lina J. Karam. "Morphological text extraction from images." *IEEE Transactions on Image Processing* 9.11
(2000): 1978-1983.

[9] Hoang, T. V., & Tabbone, S. (2010, June). Text extraction from graphical document images using sparse representation.
In *Proceedings of the 9th IAPR international workshop on document analysis systems* (pp. 143-150). ACM.

[10] Hasan, Yassin MY, and Lina J. Karam. "Morphological text extraction from images." *IEEE Transactions on Image Processing* 9.11 (2000): 1978-1983.

[11] Liu, Xiaoqing, and Jagath Samarabandu. "Multiscale edge-based text extraction from complex images." *2006 IEEE International Conference on Multimedia and Expo*. IEEE, 2006.