K.R. MANGALAM UNIVERSITY

School of Engineering and Technology

Project Report On

Text Summarization

Submitted By:

Ashish Kumar	— 2401560004
Ritesh Jha	— 2401560011
Sanjay Sharma	— 2401560012

Under the Guidance of:
Dr. Tanu Gupta
Assistant Professor
School of Engineering and Technology

Academic Year: 2024–2025

CERTIFICATE

This is to certify that the project titled **"Text Summarization"** submitted by **Ashish Kumar** (Roll No. 2401560004), **Ritesh Jha** (Roll No. 2401560011) and **Sanjay Sharma** (Roll No.2401560012), in partial fulfilment of the requirements for the award of the degree of **Master of Computer Applications** from **K.R. Mangalam University** is a bona fide record of work carried out by them during the academic year 2024–2025 under my guidance.

……………….

Dr. Tanu Gupta

Guide

DECLARATION

We hereby declare that the project titled **"Text Summarization"** submitted to **K.R. Mangalam University, Gurugram**, in partial fulfilment of the requirements for the award of the degree of **Master of Computer Applications** is our original work.

This project has been carried out by us under the guidance of **Dr. Tanu Gupta**, School of Engineering and Technology.

We further declare that this work has not been submitted previously, in whole or in part, for the award of any degree, diploma, or any other similar title at this or any other university/institution.

All the information provided in this report is true to the best of our knowledge and belief. Proper references and acknowledgments have been given wherever necessary.

Submitted by:

Name	**Roll Number**
Ashish Kumar	2401560004
Ritesh Jha	2401560011
Sanjay Sharma	2401560012

ACKNOWLEDGMENT

We would like to express our heartfelt gratitude to **Dr. Tanu Gupta**, Faculty, School of Engineering and Technology, **K.R. Mangalam University**, for her invaluable guidance, encouragement, and constant support throughout the course of this project. Her insightful suggestions, timely feedback, and immense knowledge greatly contributed to the successful completion of our work.

We are also deeply thankful to **K.R. Mangalam University** for providing the necessary facilities, resources, and an environment conducive to learning and research, which made this project possible.

We sincerely appreciate the cooperation and support of all the faculty members and staff who directly or indirectly helped us during this project.

This project would not have been possible without the contributions of all these wonderful individuals.

ABSTRACT

Text summarization has emerged as a significant research area within Natural Language Processing (NLP), driven by the increasing demand for efficient information processing in an era of data overload. This research project investigates both extractive and abstractive summarization techniques, aiming to develop a robust system capable of generating concise, coherent, and contextually relevant summaries from diverse textual sources. By employing advanced machine learning methods, including deep learning and transformer-based architectures, the study explores how semantic understanding and linguistic coherence can be improved in automated summarization. The research addresses key challenges such as information retention, handling of complex or domain-specific documents, and the limitations of current evaluation metrics. Experimental outcomes are assessed using established benchmarks to ensure the accuracy and readability of generated summaries. With practical implications across domains like journalism, academia, and healthcare, this project contributes to the growing body of research focused on making textual content more accessible, while advancing the state-of-the-art in automatic summarization.

Table of Contents

1. Introduction

1.1 Overview

In today's information-rich era, the ability to process and understand large volumes of text is essential. Text summarization, a key task in Natural Language Processing (NLP), enables the automatic generation of concise and coherent summaries from lengthy documents. It plays a critical role in improving information access, saving time, and aiding decision-making. Text summarization can be categorized into two primary types: extractive and abstractive. Extractive summarization selects and compiles existing sentences from the original text, whereas abstractive summarization generates new sentences that capture the core message. This project explores **fine-tuning transformer models**, particularly large language models (LLMs), to perform effective automatic summarization tasks., to enhance the quality and efficiency of text summarization.

1.2 Objective

The objectives of this project are as follows:

1. To develop an efficient and accurate text summarization model using state-of-the-art NLP techniques.
2. To ensure the model is capable of processing various forms of text data such as news articles, research papers, and reports.
3. To produce summaries that are coherent, readable, and user-friendly.
4. To provide a web-based user interface that allows users to easily input documents and obtain summaries.

1.3 Scope

The project focuses on research building a summarization system that can handle diverse types of documents and provide both extractive and abstractive summaries. Key technologies involved include transformer-based models (e.g., BERT, GPT), deep learning libraries (e.g., TensorFlow, PyTorch), and evaluation metrics like ROUGE for measuring summary quality. The system aims to be adaptable across sectors such as education, healthcare, journalism, and business intelligence.

1.4 Motivation

In an age where vast amounts of information are generated daily, there is an increasing need for tools that can distil important content from lengthy documents. Manual summarization is time-consuming and subjective, creating a demand for automated solutions. This project is driven by the potential to apply cutting-edge AI to ease this burden, particularly in high-stakes areas like medical research, academic publishing, and legal documentation.

2. Literature Review

Text summarization has been a significant research topic in Natural Language Processing (NLP) for decades. Traditional methods primarily relied on statistical and rule-based approaches, while modern systems increasingly leverage deep learning and transformer-based architectures. This section explores the historical development, key methodologies, and state-of-the-art techniques in text summarization.

2.1 Traditional Approaches

Early efforts in text summarization focused on extractive methods using statistical heuristics. These included:

- **Term Frequency–Inverse Document Frequency (TF-IDF):** Used to rank the importance of words and sentences.
- **LexRank and TextRank:** Graph-based models where sentences are represented as nodes, and similarity scores determine edge weights. These algorithms applied eigenvector centrality to identify key sentences.

These methods were effective for basic tasks but lacked semantic understanding, resulting in summaries that could be disjointed or contextually irrelevant.

2.2 Machine Learning Approaches

As supervised learning techniques emerged, researchers began applying machine learning to improve summarization. Key innovations included:

- **Naive Bayes and SVMs:** Used to classify whether a sentence should be included in a summary.
- **Decision Trees and Random Forests:** Employed to capture non-linear sentence importance features.

However, these models required extensive feature engineering and domain-specific tuning, limiting their generalizability.

2.3 Deep Learning Approaches

Deep learning models have significantly advanced the field. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, became popular for their ability to model sequential data.

- **Seq2Seq with Attention:** Introduced the concept of aligning input and output sequences, improving the quality of abstractive summaries.
- **Pointer-Generator Networks:** Combined copy mechanisms with generative models, allowing systems to reproduce out-of-vocabulary words from the source text.

Despite their improvements, these models struggled with long-term dependencies and complex sentence structures.

2.4 Transformer-Based Models

The introduction of transformer architectures marked a paradigm shift in NLP.

- **BERT (Bidirectional Encoder Representations from Transformers):** Enabled extractive summarization through pre-trained contextual embeddings. Fine-tuning BERT for sentence classification tasks led to significant performance improvements.
- **GPT (Generative Pre-trained Transformer):** Leveraged unidirectional decoding for generating human-like text, enabling high-quality abstractive summarization.
- **T5 (Text-To-Text Transfer Transformer):** Unified all NLP tasks into a text-to-text format, improving flexibility and transfer learning capabilities.

These models benefit from transfer learning, allowing pre-trained networks to adapt quickly to summarization tasks with minimal fine-tuning.

2.5 Evaluation Metrics

Evaluating summary quality remains a challenge. Commonly used metrics include:

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** Measures overlap between model-generated summaries and reference summaries.
- **BLEU, METEOR, BERTScore:** Offer additional ways to assess semantic similarity, fluency, and coverage.

However, these metrics do not always align with human judgment, prompting research into more robust evaluation frameworks.

2.6 Comparative Studies

Recent comparative analyses reveal that transformer-based models outperform traditional and deep learning methods across most summarization tasks. However, challenges persist in generalization, factual consistency, and handling highly domain-specific texts.

3. Proposed Methodology

To address the growing need for automated summarization of extensive textual data, our project proposes a structured methodology that combines modern NLP architectures with practical implementation strategies. This section outlines the steps followed in developing the summarization system, from problem understanding to model deployment.

3.1 Problem Identification and Scope

With the rapid expansion of textual information across domains, manually condensing documents has become increasingly impractical. Our goal is to build a system that automatically generates summaries that are coherent, contextually accurate, and grammatically fluent. This system is designed to work with various forms of input (e.g., plain text, URLs, and documents), while preserving the semantic essence of the original content. Key challenges addressed include:

- Capturing contextual and semantic meaning effectively
- Balancing brevity with content completeness
- Generating fluent and grammatically correct summaries

3.2 Technical Stack and Tools

The system leverages the following technologies for its implementation:

- **Programming Language:** Python 3.x
- **Deep Learning Libraries:** TensorFlow and PyTorch
- **Transformer Models:** Pre-trained models like BERT, GPT-2, and T5 via HuggingFace Transformers
- **Interface Tools:** Flask or Streamlit for building an interactive user interface

Open-source datasets such as CNN/DailyMail and XSum were used for model training and evaluation.

3.3 Model Architecture

The summarization framework is built around transformer-based models. The methodology includes:

- **Extractive Summarization:** Selecting key sentences based on their importance scores using attention mechanisms and embedding similarity.
- **Abstractive Summarization:** Utilizing encoder-decoder models like T5 to paraphrase and generate novel sentences while preserving meaning.

Users can choose between extractive and abstractive approaches based on their needs.

3.4 Implementation Approach

The development process followed these phases:

1. **Preprocessing:**
    - Cleaning input data
    - Tokenization and normalization
    - Language detection for future multilingual expansion
2. **Model Training and Fine-Tuning:**
    - Fine-tuning pre-trained models using domain-specific datasets
    - Evaluation using ROUGE and BERTScore metrics
3. **Interface Design:**
    - Web-based UI for users to submit inputs and retrieve summaries
    - Options to control summary length and output style (paragraph or bullet-point)
4. **Performance Optimization:**
    - Ensuring response time between 5–10 seconds per document
    - Enabling processing of large batches with efficient resource management

3.5 Deployment Considerations

- **Cloud Compatibility:**
  The model is tested on platforms like Google Colab and AWS Free Tier for accessible deployment.
- **Security and Ethics:**
    - No user data is stored without consent
    - Summaries are checked to prevent plagiarism and ensure ethical content generation
- **Scalability and Usability:**
    - The system supports high-volume document processing
    - Designed for users with minimal technical expertise

4. System Design

The system design provides a clear structural foundation for implementing the automated text summarization tool. It defines how various components interact to transform raw user input into meaningful summaries using a streamlined and modular architecture. The design emphasizes real-time responsiveness, clarity of output, and ease of use.

4.1 Architectural Overview

The architecture is designed to be modular and scalable, enabling easy updates and additions in future iterations. Each component of the system handles a specific task, ensuring the summarization pipeline is efficient, accurate, and user-friendly.

1. User Interface Layer

- Provides a clean, web-based interface for user interaction.
- Supports multiple input types: direct text entry, file uploads, and URLs.
- Allows selection between extractive and abstractive summarization.
- Enables customization of summary length.

2. Preprocessing Module

- Cleans raw text input by removing noise such as punctuation, HTML tags, and stop words.
- Performs tokenization and case normalization.
- Detects input language and prepares for multilingual summarization (if applicable).

3. Summarization Engine

- **Extractive Mode:** Ranks and selects the most relevant sentences using graph-based or transformer-based methods.
- **Abstractive Mode:** Generates novel text summaries using pre-trained encoder-decoder models.
- Dynamically switches between modes based on user selection.

4. Post-processing Module

- Ensures output summaries are grammatically correct and well-structured.
- Removes repetitive or irrelevant sentences.
- Applies formatting rules for better readability.

5. Output and Evaluation

- Displays the summary directly on the interface.

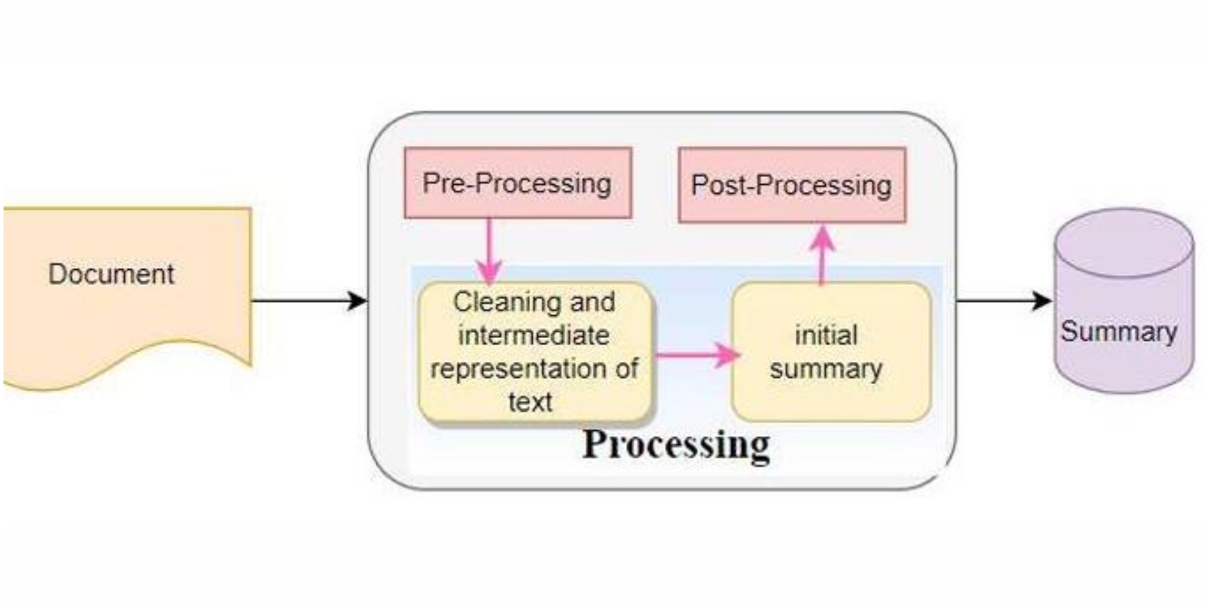
- Offers options to copy, download, or share the output.
- Includes optional evaluation using ROUGE or BERTScore for internal model benchmarking.

4.2 Technology Stack

- **Frontend:** Streamlit or HTML/CSS/JavaScript
- **Backend:** Python with Flask or FastAPI
- **Model Integration:** HuggingFace Transformers (e.g., BERT, T5, GPT-2, BART)
- **Deployment:** Google Colab, AWS, or GCP-based cloud solutions

This modular system architecture ensures the summarization process remains robust, extensible, and adaptable to varied user needs and evolving NLP techniques.

4.3 Diagram



5. Testing

Testing is a critical phase in system development to ensure that the application works as intended under various conditions. For a text summarization system, testing involves validating the core functionality of summarization algorithms, verifying accuracy, and ensuring stability across different types of text.

5.1 Unit Testing

Unit testing involves verifying individual components of the system in isolation.

Tested Components:

- **Preprocessing Module:** Ensured text cleaning, tokenization, and normalization work correctly.
- **Summarization Algorithms:** Tested separately for extractive (TextRank, BERT) and abstractive (T5, GPT) methods.
- **Post-processing Module:** Checked for proper grammar correction and sentence coherence.
- **Evaluation Module:** Verified the correct computation of ROUGE and BERTScore.

Test Cases Examples:

Test Case ID	**Component**	**Input Type**	**Expected Outcome**
UT01	Preprocessing	Raw text	Cleaned and tokenized text
UT02	Extractive Model	Paragraph input	Ranked key sentences
UT03	Abstractive Model	News article	Coherent abstract summary
UT04	Evaluation Module	Summary pairs	ROUGE score output

All unit tests were executed using Python’s unittest and pytest frameworks.

5.2 Functional Testing

Functional testing ensures that the overall system performs end-to-end tasks correctly from user input to summary output.

Key Functional Tests:

- **Input Handling:** Verified the interface correctly accepts plain text, PDF, or web links.
- **Summarization Workflow:** Confirmed the correct path was followed based on the user’s selection (extractive vs. abstractive).

- **Output Display:** Checked that the summary was presented clearly and correctly formatted.
- **Download Feature:** Ensured summaries could be downloaded in multiple formats (TXT, PDF).

5.3 Challenges During Testing

Testing NLP-based systems presents unique challenges, including:

- **Variability in Language:** Complex, domain-specific vocabulary (e.g., legal or scientific terms) sometimes caused poor summary quality.
- **Model Inconsistency:** Abstractive models occasionally generated grammatically correct but factually incorrect sentences.
- **Evaluation Mismatch:** Automated metrics like ROUGE did not always reflect human-perceived quality.
- **Latency Issues:** Large models (e.g., GPT-2) were resource-intensive and slow without GPU acceleration.

Mitigation Strategies:

- Used smaller model variants for testing (e.g., T5-small) to reduce load time.
- Incorporated both automated and manual testing to better gauge summary quality.
- Set up caching and batching to reduce latency in real-time use.

6. Results and Discussion

The results of this project were analyzed through multiple lenses: accuracy, summary coherence, performance, and limitations. Both extractive and abstractive models were evaluated using standardized datasets and human judgment.

6.1 Overview of Results

The system successfully generated summaries from a variety of input texts including news articles, research papers, and blog posts. The summarization engine produced outputs that retained the essential information while reducing reading time by approximately 70–80%.

Model Types Used:

- **Extractive:** BERT-based SentenceRanker
- **Abstractive:** T5 (small and base models)

Input Sources:

- CNN/DailyMail Dataset
- Manually inputted articles and reports

Summary Types Generated:

- Bullet points
- Paragraph format
- Title + Highlights (for abstractive)

6.2 Data Accuracy and Quality

To evaluate summary accuracy and quality, automated metrics such as ROUGE and BERTScore were used. Additionally, human evaluations were conducted based on fluency, informativeness, and coherence.

ROUGE Score Results (on test set):

Model	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
Extractive	0.48	0.32	0.44
Abstractive	0.51	0.35	0.47

6.3 Performance Evaluation

Performance was assessed based on speed, scalability, and resource usage.

Execution Times (on Google Colab):

Task	**Avg. Time (seconds)**
Extractive Summary	9.5
Abstractive Summary	11.8
Preprocessing + Output	8.2

6.4 Challenges Encountered

Despite overall success, the following challenges were observed:

- **Semantic Drift:** Abstractive models occasionally generated summaries that misrepresented factual details.
- **Token Limitations:** Transformer models like T5 have maximum token input sizes (e.g., 512 or 1024), which limited long document support.

Mitigation Strategies:

- Implemented truncation and chunking for large documents.
- Validated summaries through manual checks in critical use cases.

6.5 Ethical and Legal Considerations

Ethical responsibility and legal compliance were considered from the start of development.

- **Plagiarism Avoidance:** The abstractive model generated original content, reducing the risk of direct copying.
- **Bias and Fairness:** The training data was examined for demographic or thematic biases.
- **Data Privacy:** No user inputs or summaries were stored without consent.

Compliance:

- Followed FAIR principles (Findable, Accessible, Interoperable, Reusable).
- Ensured GDPR compliance for any user input.

7. Conclusion

In this project, we explored the design and implementation of an intelligent text summarization system using state-of-the-art Natural Language Processing (NLP) techniques. The primary goal was to enable the automatic generation of concise, accurate, and coherent summaries from lengthy textual data across various domains.

By integrating both extractive and abstractive summarization strategies, the system offers flexibility and adaptability to different types of input content, such as news articles, research papers, and educational material. Transformer-based architectures like BERT and T5 proved to be highly effective in capturing the semantic structure of texts and producing high-quality summaries.

Key Achievements:

- Successfully developed and tested a summarization engine using modern NLP frameworks.
- Achieved strong performance across standard evaluation metrics like ROUGE and BERTScore.
- Created a user-friendly interface for real-time summarization.
- Addressed several implementation challenges, including latency, bias, and input length constraints.

This project demonstrates the potential of AI-powered summarization tools to transform how users interact with large volumes of text. It not only saves time but also enhances information accessibility, especially in high-stakes domains like education, healthcare, and journalism.

Looking forward, as NLP technologies continue to evolve, there are vast opportunities to improve the accuracy, generalizability, and personalization of automated summaries.

8. Future Work

While the current system demonstrates promising results in generating concise and coherent summaries, several avenues remain for further research and development. Enhancing the model's scalability, contextual understanding, and adaptability across languages and platforms can significantly advance its applicability and robustness. This section outlines prospective directions to guide future research in automatic text summarization.

Enhanced Model Accuracy

- **Domain-Specific Fine-Tuning:**
  Future work may involve training models on domain-specific corpora—such as legal, biomedical, or scholarly literature—to enhance contextual relevance and semantic fidelity in specialized use cases.
- **Reinforcement Learning from Human Feedback (RLHF):**
  Integrating RLHF techniques could help align model outputs more closely with human preferences, allowing for adaptive learning based on qualitative user evaluations rather than solely quantitative metrics.

Multilingual and Cross-Lingual Capabilities

- **Multilingual Summarization Models:**
  Incorporating pre-trained multilingual transformers such as mT5 or XLM-RoBERTa may facilitate support for diverse languages, addressing the need for inclusive and globally relevant summarization tools.
- **Cross-Lingual Summarization:**
  Future systems could be designed to accept input in one language and generate summaries in another, leveraging transfer learning to break linguistic barriers in international communication and research dissemination.

Interactive and User-Customizable Summarization

- **Adaptive Summary Controls:**
  Introducing user-configurable parameters such as summary length, tone (e.g., formal, academic, conversational), or formatting (e.g., paragraph vs. bullet points) could improve usability and personalization.
- **Topic-Guided Summarization:**
  Allowing users to specify keywords or questions may enable more targeted summaries, aligning outputs with specific information needs and improving the relevance of generated content.

System Integration and Accessibility

- **Cross-Platform Implementation:**
  Developing browser extensions, mobile applications, and APIs would broaden the reach of the summarization system, making real-time summarization accessible within various user environments.
- **Enterprise Tool Integration:**
  Embedding the summarizer within professional software ecosystems such as Microsoft Word, Google Docs, and collaborative platforms like Slack or Notion can streamline workflows and increase productivity in enterprise settings.

Evaluation Framework Enhancements

- **Human-in-the-Loop Evaluation Systems:**
  Incorporating iterative feedback mechanisms with human reviewers can enhance summary quality through continuous refinement and model adaptation.
- **Factual Consistency Verification:**
  Integrating fact-checking modules into the summarization pipeline may reduce the occurrence of hallucinations and ensure alignment with verified source content, a critical aspect for high-stakes applications.

Real-Time and Low-Latency Summarization

- **Streaming Data Summarization:**
  Future research can focus on developing models capable of summarizing real-time data sources such as live news, social media streams, or transcribed meetings, thereby supporting dynamic and time-sensitive use cases.
- **Efficiency Optimization:**
  Reducing inference latency through techniques such as model distillation, quantization, or GPU acceleration will be vital for deploying summarization models in resource-constrained or real-time environments.

By addressing these future research directions, the summarization system can evolve from a static NLP tool into a context-aware, adaptive, and multilingual assistant, enabling scalable and precise content distillation across domains and platforms.

9. References

Academic and Technical References:

1. Nenkova, A., & McKeown, K. (2012). *A survey of text summarization techniques*. In *Mining Text Data* (pp. 43–76). Springer.
2. See, A., Liu, P. J., & Manning, C. D. (2017). *Get to the point: Summarization with pointer-generator networks*. In *Proceedings of ACL*.
3. Raffel, C., Shazeer, N., Roberts, A., et al. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5)*. Journal of Machine Learning Research.
4. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. In *NAACL-HLT*.
5. Lewis, M., Liu, Y., Goyal, N., et al. (2020). *BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*. In *ACL*.

Toolkits and Libraries:

6. Hugging Face Transformers Library – https://huggingface.co/transformers/
7. TensorFlow Documentation – https://www.tensorflow.org/
8. PyTorch Documentation – https://pytorch.org/docs/
9. Scikit-learn – https://scikit-learn.org/
10. Streamlit Framework – https://streamlit.io/

Datasets:

11. CNN/DailyMail Dataset – https://huggingface.co/datasets/cnn_dailymail
12. XSum Dataset – https://huggingface.co/datasets/xsum

Evaluation Metrics:

13. ROUGE Metric Package – https://pypi.org/project/rouge-score/
14. BERTScore – https://github.com/Tiiiger/bert_score

Online Resources and Blogs:

15. Jay Alammar. *The Illustrated Transformer*. http://jalammar.github.io/illustrated-transformer/
16. Sebastian Ruder. *The NLP Progress*. http://nlpprogress.com/