
# PROJECT REPORT

# ON

# MASKED WORD FINDER

## SUBMITTED TO

## DEPARTMENT OF COMPUTER SCIENCE

## UNDER THE SUPERVISION OF

## Dr. Atul Garg

**Submitted By: Vidhyun Kapoor**
**Roll No: 1810991180**

**CHITKARA UNIVERSITY**
**CHANDIGARH-PATIALA NATIONAL HIGHWAY, RAJPURA, DISTT. PATIALA, PUNJAB, INDIA**

# CONTENTS

# DECLARATION

I hereby declare that the project work titled, **"Masked Word Finder"** submitted as part of Bachelor's degree in CSE, at Chitkara University, Punjab, is an authentic record of my own work carried out under the supervision of Dr. Atul Garg.



**Signature(s):**

# ACKNOWLEDGEMENT



I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend a sincere thanks to all of them.

I am highly indebted to Dr. Atul Garg for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.
I would like to express my gratitude towards my parents & members of CSE Department for their kind co-operation and encouragement which helped in completion of this project.

I would like to express special gratitude and thanks to industry persons for giving me such attention and time.

# List of Figures and Tables

# Project Title: Masked Word Finder



**Introduction:** In today's world, people have a lot of work which requires strong vocabulary of English language along with a decent grip on grammar. Some people tend to struggle while finding the correct word required to form a sentence to make a better understanding for everyone. The aim of this project is preparing a platform that can help the users by suggesting meaningful words in between any sentence they are struggling to form.

**Project Category:** In my view this project belongs to Literary Assistance because this project is for providing a helping hand to anyone who wishes to improve vocabulary or use correct words grammatically. It will be very beneficial since it'll provide multiple suggestions of words wherever the user requires no matter how long the entire text input is.

**Abstract:** In today's world people have no time as they want their work to be done as soon as possible. Same is in the case of preparing projects, writing essays, presentations etc. with a lot of technical terms that also requires a strong English vocabulary. As they do not want to search the correct word internet multiple time which might take hours or minutes, this is the project which can help in solving this issue and saving a lot of time. So, the main purpose of this project is to build a software that allows the user to enter a text input and search for the most appropriate word wherever they get stuck.



## Literature Review:

Word prediction is a research area where a very challenging and ambitious task is faced, basically with methods coming from Artificial Intelligence, Natural Language Processing and Machine Learning.

The main goal of word prediction is guessing and completing the word a user is willing to type. Word predictors are intended to support writing and are commonly used in combination with assistive devices such as keyboards, virtual keyboards, touchpads and pointing devices. Another potential application is in text-entry interfaces for messaging on mobile phones and typing on handheld and ubiquitous devices (e.g. PDAs or smartphones).

Prediction methods have become quite known as largely adopted in mobile phones and PDAs, where multitap is the input method. Nuance T9 (formerly Tegic Communications T9)l and Zi Corporation eZiText2 are commercial systems that adopt a very simple method of prediction based on dictionary disambiguation. At each user keystroke the system selects the letter between the ones associated with the key guessing it from a dictionary of words: hence they are commonly referred to as letter predictors. Letter predictors bring a Keystroke Saving (KS) but it has been proven to be not completely free from ambiguities that are more frequent for inflected languages. So, it is not surprising that these methods had a great success for non-inflected

languages such as English: the limited number of inflectional forms lead to very high KS that, at the moment, are above 40%.

Word prediction is a more sophisticated technique within recent research. Differently from letter predictors, word predictors typically make use of language modelling techniques, namely stochastic models that are able to give context information in order to improve the prediction quality.



The language that the system has to model influences the prediction techniques; inflected languages pose a harder challenge to prediction algorithms, since they have to deal with a usually high number of inflected forms that dramatically decrease Keystroke Saving. To simplify the task of predicting the correct form, some techniques provide a two-step procedure, choosing first only among word "roots", and proposing all the possible word forms only when the user selects a root. FastType relies instead on Part-of-Speech (POS) and related morpho-syntactic information to provide a one-step procedure, presenting to the user a list of word forms. This procedure, combined with on-the-fly POS tagging, enables FastType to boost performances, cutting off of the prediction list all words whose gender, number, tense or mood are not consistent with the sentence context. The prediction list becomes also a "guide tool" to write syntactically correct sentences.

# Work Done:

**Overview:** It is a web page that has a user-friendly interface. User can enter text along with the position of the missing word. He/she can also choose how many words should be predicted for the missing space so it'll be easier to pick from multiple options. It hardly takes few seconds for the NLP frameworks to process the text and show outputs. The main purpose of building this is that some people do not have time to search on the internet and browse through multiple websites to find the best word as per the sentence formation.



## Purpose:

1. To provide a word finding platform service to save time.
2. User can find the word on any position.
3. Multiple frameworks to increase the accuracy.
4. To ensure that the correct word is predicted, multiple outputs can be generated.
5. A user can customize the platform in his/her desired way.
6. Quick prediction will be there, no need to wait for the results for so long.

## Overall Description:

### 1. What is NLP ?



Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning, complete with the speaker or writer's intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real-time. There's a good chance you've interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

### 2. History of NLP

In the early 1900s, a Swiss linguistics professor named Ferdinand de Saussure died, and in the process, almost deprived the world of the concept of "Language as a Science." From 1906 to 1911, Professor Saussure offered three courses at the University of Geneva, where he developed an approach describing languages as

"systems." Within the language, a sound represents a concept – a concept that shifts meaning as the context changes.

He argued that meaning is created inside language, in the relations and differences between its parts. Saussure proposed "meaning" is created within a language's relationships and contrasts. A shared language system makes communication possible. Saussure viewed society as a system of "shared" social norms that provides conditions for reasonable, "extended" thinking, resulting in decisions and actions by individuals. (The same view can be applied to modern computer languages).



Saussure died in 1913, but two of his colleagues, Albert Sechehaye and Charles Bally, recognized the importance of his concepts. (Imagine the two, days after Saussure's death, in Bally's office, drinking coffee and wondering how to keep his discoveries from being lost forever). The two took the unusual steps of collecting "his notes for a manuscript," and his students' notes from the courses. From these, they wrote the *Cours de Linguistique Générale*, published in 1916. The book laid the foundation for what has come to be called the structuralist approach, starting with linguistics, and later expanding to other fields, including computers.

In 1950, Alan Turing wrote a paper describing a test for a "thinking" machine. He stated that if a machine could be part of a conversation through the use of a teleprinter, and it imitated a human so completely there were no noticeable differences, then the machine could be considered capable of thinking. Shortly after this, in 1952, the Hodgkin-Huxley model showed how the brain uses neurons in forming an electrical network. These events helped inspire the idea of Artificial Intelligence (AI), Natural Language Processing (NLP), and the evolution of computers.

Noam Chomsky published his book, *Syntactic Structures*, in 1957. In it, he revolutionized previous linguistic concepts, concluding that for a computer to understand a language, the sentence structure would have to be changed. With this as his goal, Chomsky created

a style of grammar called Phase-Structure Grammar, which methodically translated natural language sentences into a format that is usable by computers. (The overall goal was to create a computer capable of imitating the human brain, in terms of in thinking and communicating, or AI.)



In 1958, the programming language LISP (Locator/Identifier Separation Protocol), a computer language still in use today, was released by John McCarthy. In 1964, ELIZA, a "typewritten" comment and response process, designed to imitate a psychiatrist using reflection techniques, was developed. (It did this by rearranging sentences and following relatively simple grammar rules, but there was no understanding on the computer's part.) Also, in 1964, the U.S. National Research Council (NRC) created the Automatic Language Processing Advisory Committee, or ALPAC, for short. This committee was tasked with evaluating the progress of Natural Language Processing research.

In 1966, the NRC and ALPAC initiated the first AI and NLP stoppage, by halting the funding of research on Natural Language Processing and machine translation. After twelve years of research, and $20 million dollars, machine translations were still more expensive than manual human translations, and there were still no computers that came anywhere near being able to carry on a basic conversation. In 1966, Artificial Intelligence and Natural Language Processing (NLP) research were considered a dead-end by many (though not all).

## 3. Return of the NLP

It took nearly fourteen years (until 1980) for Natural Language Processes and Artificial Intelligence research to recover from the broken expectations created by extreme enthusiasts. In some ways, the AI stoppage had initiated a new phase of fresh ideas, with earlier concepts of machine translation being abandoned, and new

ideas promoting new research, including expert systems. The mixing of linguistics and statistics, which had been popular in early NLP research, was replaced with a theme of pure statistics. The 1980s initiated a fundamental reorientation, with simple approximations replacing deep analysis, and the evaluation process becomes more rigorous.



Until the 1980s, the majority of NLP systems used complex, "handwritten" rules. But in the late 1980s, a revolution in NLP came about. This was the result of both the steady increase of computational power and the shift to Machine Learning algorithms. While some of the early Machine Learning algorithms (decision trees provide a good example) produced systems similar to the old school handwritten rules, research has increasingly focused on statistical models. These statistical models are capable of making soft, probabilistic decisions. Throughout the 1980s, IBM was responsible for the development of several successful, complicated statistical models.

In the 1990s, the popularity of statistical models for Natural Language Processes analyses rose dramatically. The pure statistics NLP methods have become remarkably valuable in keeping pace with the tremendous flow of online text. N-Grams have become useful, recognizing and tracking clumps of linguistic data, numerically. In 1997, LSTM recurrent neural net (RNN) models were introduced and found their niche in 2007 for voice and text processing. Currently, neural net models are considered the cutting edge of research and development in the NLP's understanding of text and speech generation.

In 2001, Yoshio Bengio and his team proposed the first neural "language" model, using a feed-forward neural network. The feed-forward neural network describes an artificial neural network that does not use connections to form a cycle. In this type of network, the data moves only in one direction, from input nodes, through any hidden nodes, and then on to the output nodes. The feed-

forward neural network has no cycles or loops, and is quite different from the recurrent neural networks.

In the year 2011, Apple's Siri became known as one of the world's first successful NLP/AI assistants to be used by general consumers. Within Siri, the Automated Speech Recognition module translates the owner's words into digitally interpreted concepts. The Voice-Command system then matches those concepts to predefined commands, initiating specific actions. For example, if Siri asks, "Do you want to hear your balance?" it would understand a "Yes" or "No" response, and act accordingly.



By using Machine Learning techniques, the owner's speaking pattern doesn't have to match exactly with predefined expressions. The sounds just have to be reasonably close for an NLP system to translate the meaning correctly. By using a feedback loop, NLP engines can significantly improve the accuracy of their translations, and increase the system's vocabulary. A well-trained system would understand the words, "Where can I get help with Big Data?" "Where can I find an expert in Big Data?" or "I need help with Big Data," and provide the appropriate response.

The combination of a dialog manager with NLP makes it possible to develop a system capable of holding a conversation, and sounding human-like, with back-and-forth questions, prompts, and answers. Our modern AIs, however, are still not able to pass Alan Turing's test, and currently do not sound like real human beings. (Not yet, anyway.)

## 4. Tasks performed through NLP

Human language is filled with ambiguities that make it incredibly difficult to write software that accurately determines the intended meaning of text or voice data. Homonyms, homophones, sarcasm, idioms, metaphors, grammar and usage exceptions, variations in

sentence structure—these are just a few of the irregularities of human language that take humans years to learn, but that programmers must teach natural language-driven applications to recognize and understand accurately from the start if those applications are going to be useful.



Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:

- **Speech recognition**, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar.

- **Part of speech tagging**, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies ‘make’ as a verb in ‘I can make a paper plane,’ and as a noun in ‘What make of car do you own?’

- **Word sense disambiguation** is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in ‘make the grade’ (achieve) vs. ‘make a bet’ (place).

- **Named entity recognition,** or NEM, identifies words or phrases as useful entities. NEM identifies ‘Kentucky’ as a location or ‘Fred’ as a man's name.



- **Co-reference resolution** is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., ‘she’ = ‘Mary’), but it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).

- **Sentiment analysis** attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.

- **Natural language generation** is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

## 5. Tools and Approaches in NLP

### Python and the Natural Language Toolkit (NLTK)

The Python programming language provides a wide range of tools and libraries for attacking specific NLP tasks. Many of these are found in the Natural Language Toolkit, or NLTK, an open source collection of libraries, programs, and education resources for building NLP programs.

The NLTK includes libraries for many of the NLP tasks listed above, plus libraries for subtasks, such as sentence parsing, word segmentation, stemming and lemmatization (methods of

trimming words down to their roots), and tokenization (for breaking phrases, sentences, paragraphs and passages into tokens that help the computer better understand the text). It also includes libraries for implementing capabilities such as semantic reasoning, the ability to reach logical conclusions based on facts extracted from text.



### Statistical NLP, machine learning, and deep learning

The earliest NLP applications were hand-coded, rules-based systems that could perform certain NLP tasks, but couldn't easily scale to accommodate a seemingly endless stream of exceptions or the increasing volumes of text and voice data.

Enter statistical NLP, which combines computer algorithms with machine learning and deep learning models to automatically extract, classify, and label elements of text and voice data and then assign a statistical likelihood to each possible meaning of those elements. Today, deep learning models and learning techniques based on convolutional neural networks (CNNs) and recurrent neural networks (RNNs) enable NLP systems that 'learn' as they work and extract ever more accurate meaning from huge volumes of raw, unstructured, and unlabelled text and voice data sets.

## 6. NLP Use Cases

Natural language processing is the driving force behind machine intelligence in many modern real-world applications. Here are a few examples:

- **Spam detection:** You may not think of spam detection as an NLP solution, but the best spam detection technologies use NLP's text classification capabilities to scan emails for language that often indicates spam or phishing. These

indicators can include overuse of financial terms, characteristic bad grammar, threatening language, inappropriate urgency, misspelled company names, and more. Spam detection is one of a handful of NLP problems that experts consider 'mostly solved' (although you may argue that this doesn't match your email experience).



- **Machine translation:** Google Translate is an example of widely available NLP technology at work. Truly useful machine translation involves more than replacing words in one language with words of another. Effective translation has to capture accurately the meaning and tone of the input language and translate it to text with the same meaning and desired impact in the output language. Machine translation tools are making good progress in terms of accuracy. A great way to test any machine translation tool is to translate text to one language and then back to the original. An oft-cited classic example: Not long ago, translating "*The spirit is willing but the flesh is weak"* from English to Russian and back yielded "*The vodka is good but the meat is rotten*." Today, the result is "*The spirit desires, but the flesh is weak*," which isn't perfect, but inspires much more confidence in the English-to-Russian translation.

- **Virtual agents and chatbots:** Virtual agents such as Apple's Siri and Amazon's Alexa use speech recognition to recognize patterns in voice commands and natural language generation to respond with appropriate action or helpful comments. Chatbots perform the same magic in response to typed text entries. The best of these also learn to recognize contextual clues about human requests and use them to provide even better responses or options over time. The next enhancement for these applications is question answering, the ability to respond to our questions—anticipated or not—with relevant and helpful answers in their own words.

- **Social media sentiment analysis:** NLP has become an essential business tool for uncovering hidden data insights from social media channels. Sentiment analysis can analyze language used in social media posts, responses, reviews, and more to extract attitudes and emotions in response to products, promotions, and events–information companies can use in product designs, advertising campaigns, and more.

- **Text summarization:** Text summarization uses NLP techniques to digest huge volumes of digital text and create summaries and synopses for indexes, research databases, or busy readers who don't have time to read full text. The best text summarization applications use semantic reasoning and natural language generation (NLG) to add useful context and conclusions to summaries.

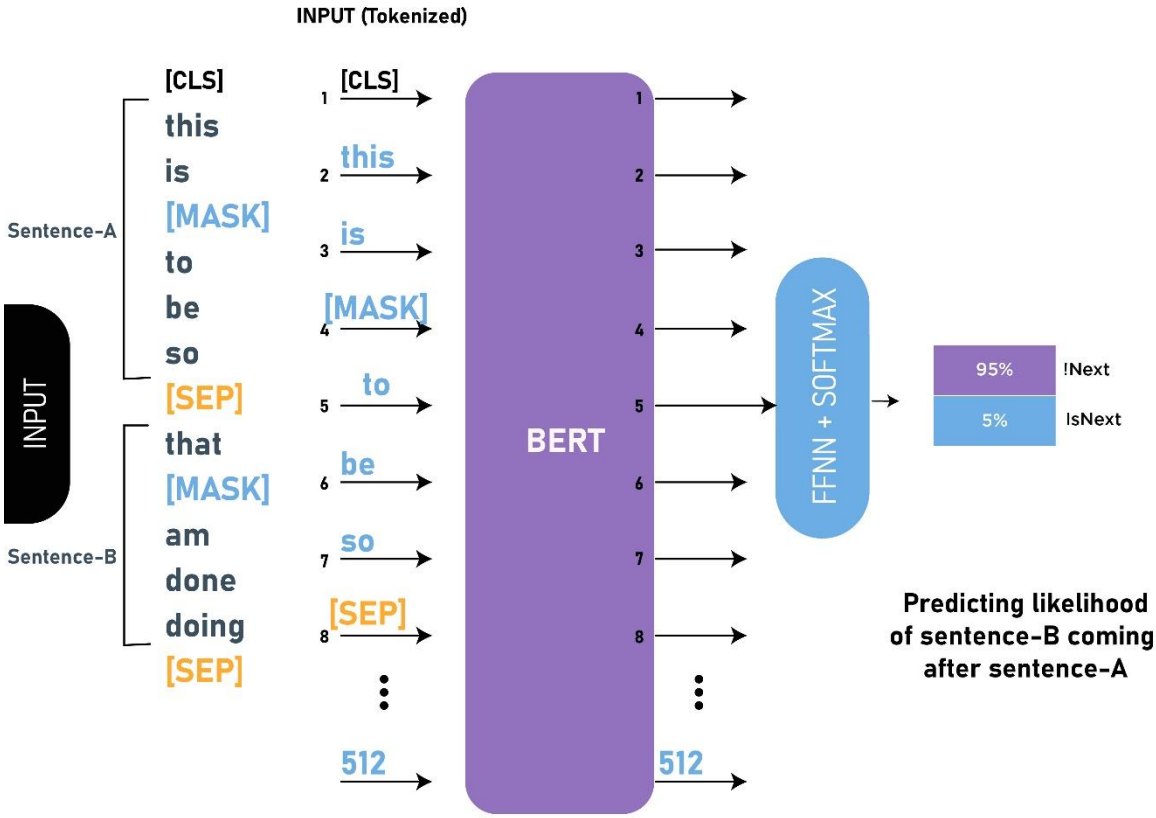
## 7. Frameworks in NLP

- **BERT:** BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others. BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel

technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

**Figure 1:** Working of the BERT framework in the project



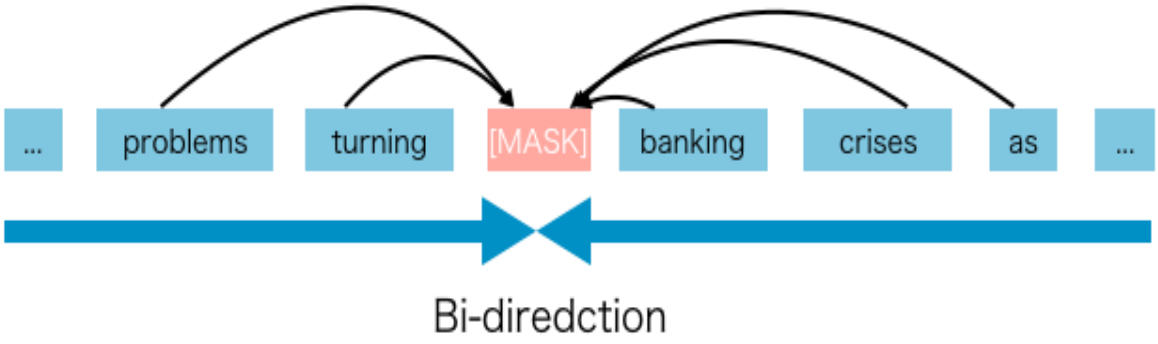



- **XLNET:** With the capability of modelling bidirectional contexts, denoising autoencoding based pretraining like BERT achieves better performance than pretraining approaches based on autoregressive language modelling. However, relying on corrupting the input with masks, BERT neglects dependency between the masked positions and suffers from a pretrain-finetune discrepancy. In light of these pros and cons, we propose XLNet, a generalized autoregressive pretraining method that enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization

order and overcomes the limitations of BERT thanks to its autoregressive formulation. Furthermore, XLNet integrates ideas from Transformer-XL, the state-of-the-art autoregressive model, into pretraining. Empirically, under comparable experiment settings, XLNet outperforms BERT on 20 tasks, often by a large margin, including question answering, natural language inference, sentiment analysis, and document ranking.



**Figure 2:** How XLNET outperforms the BERT framework in the project

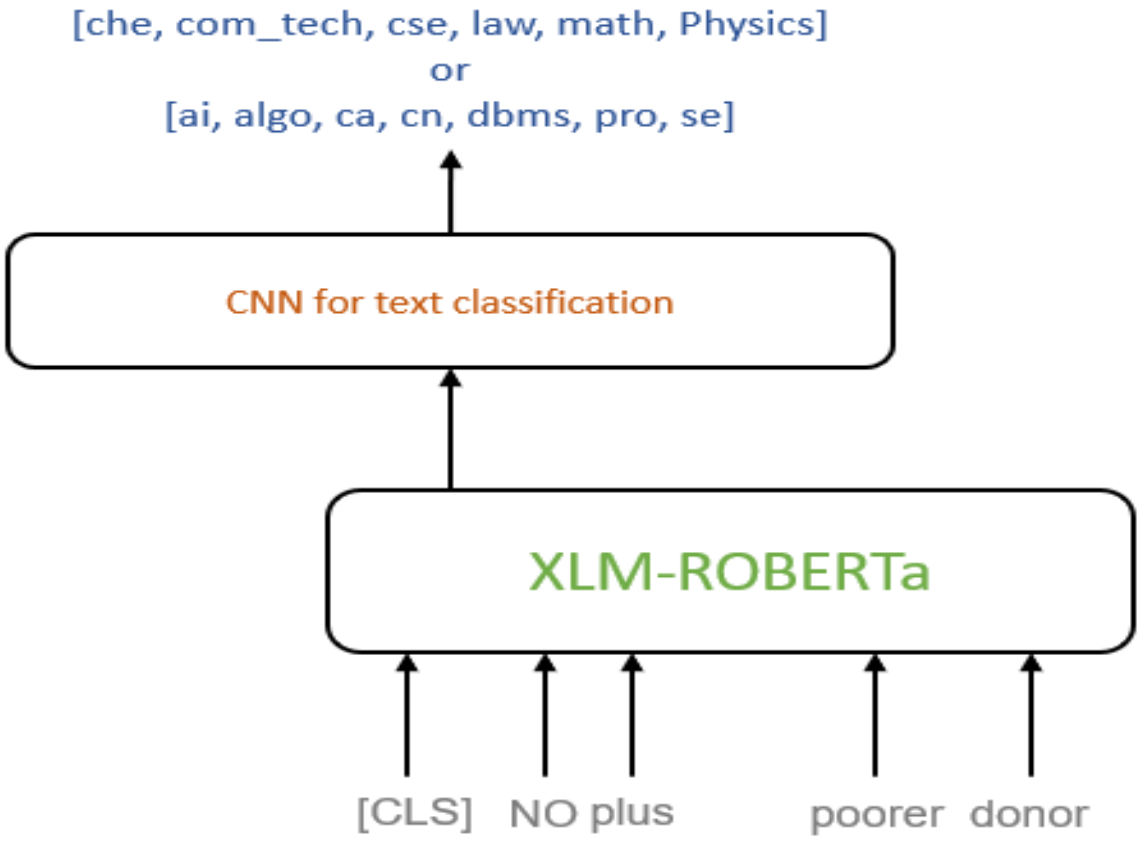



- **XLM-Roberta:** The Facebook AI team released XLM-Roberta in November 2019 as an update to their original XLM-100 model. Both are transformer-based language models, both rely on the Masked Language Model objective and both are capable of processing text from 100 separate languages. The biggest update that XLM-Roberta offers over the original is a significantly increased amount of training data. The cleaned CommonCrawl data that it is trained on takes up a whopping 2.5tb of storage! It is several orders of magnitude larger than the Wiki-100 corpus that was used to train its predecessor and the scale-up is particularly noticeable in the lower resourced languages. The “RoBERTa” part comes from the fact that its

training routine is the same as the monolingual RoBERTa model, specifically, that the sole training objective is the Masked Language Model. There is no Next Sentence Prediction á la BERT or Sentence Order Prediction á la ALBERT.



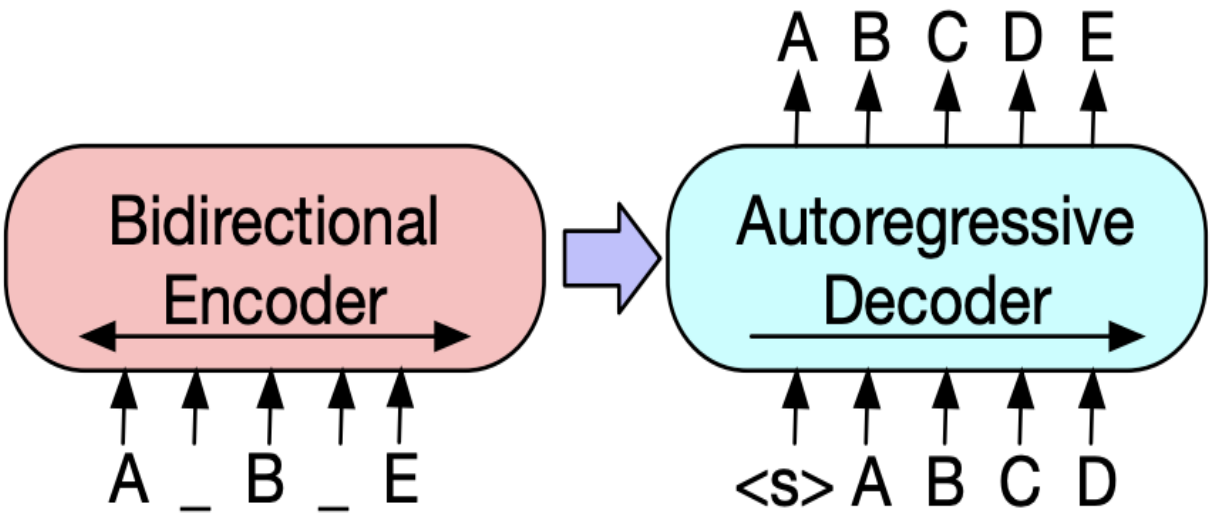
**Figure 3:** Different prediction approach of XLM-Roberta




- **BART:** We present BART, a denoising autoencoder for pretraining sequence-to-sequence models. BART is trained by corrupting text with an arbitrary noising function, and learning a model to reconstruct the original text. It uses a standard Transformer-based neural machine translation architecture which, despite its simplicity, can be seen as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right

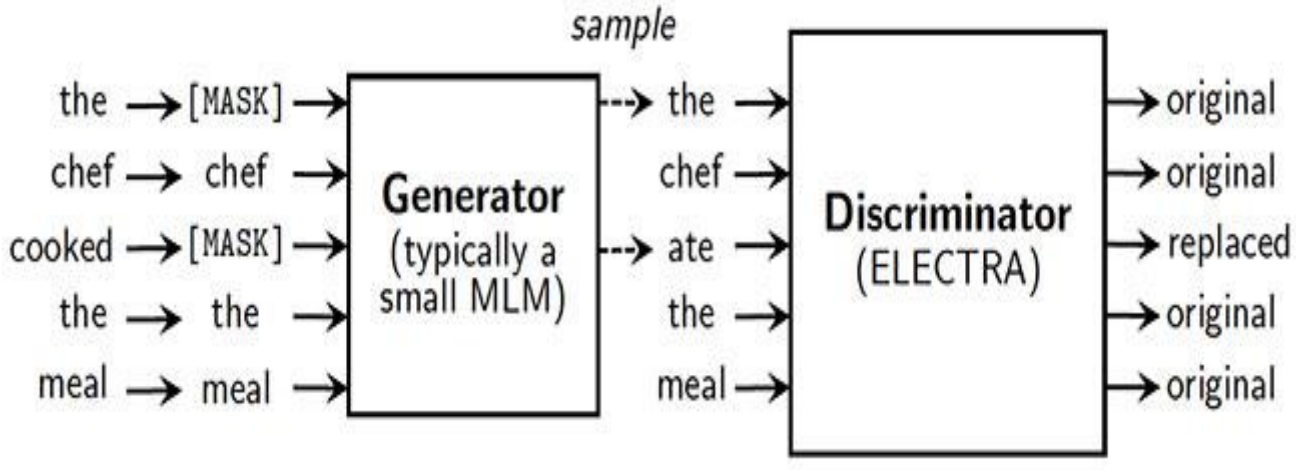
decoder), and many other more recent pretraining schemes. We evaluate a number of noising approaches, finding the best performance by both randomly shuffling the order of the original sentences and using a novel in-filling scheme, where spans of text are replaced with a single mask token. BART is particularly effective when fine-tuned for text generation but also works well for comprehension tasks. It matches the performance of RoBERTa with comparable training resources on GLUE and SQuAD, achieves new state-of-the-art results on a range of abstractive dialogue, question answering, and summarization tasks, with gains of up to 6 ROUGE. BART also provides a 1.1 BLEU increase over a back-translation system for machine translation, with only target language pretraining. We also report ablation experiments that replicate other pretraining schemes within the BART framework, to better measure which factors most influence end-task performance.

**Figure 4:** Sequence to sequence prediction method through BART



- **ELECTRA:** Masked language modelling (MLM) pre-training methods such as BERT corrupt the input by replacing some tokens with [MASK] and then train a model to reconstruct the original tokens. While they produce good results when transferred to downstream NLP tasks, they generally require large amounts of compute to be effective. As an alternative, we propose a more sample-efficient pre-training task called replaced token detection. Instead of masking the input, our approach corrupts it by replacing some tokens with plausible alternatives sampled from a small generator network. Then, instead of training a model that predicts the original identities of the corrupted tokens, we train a discriminative model that predicts whether each token in the corrupted input was replaced by a generator sample or not. Thorough experiments demonstrate this new pre-training task is more efficient than MLM because the task is defined over all input tokens rather than just the small subset that was masked out. As a result, the contextual representations learned by our approach substantially outperform the ones learned by BERT given the same model size, data, and compute. The gains are particularly strong for small models; for example, we train a model on one GPU for 4 days that outperforms GPT (trained using 30x more compute) on the GLUE natural language understanding benchmark.

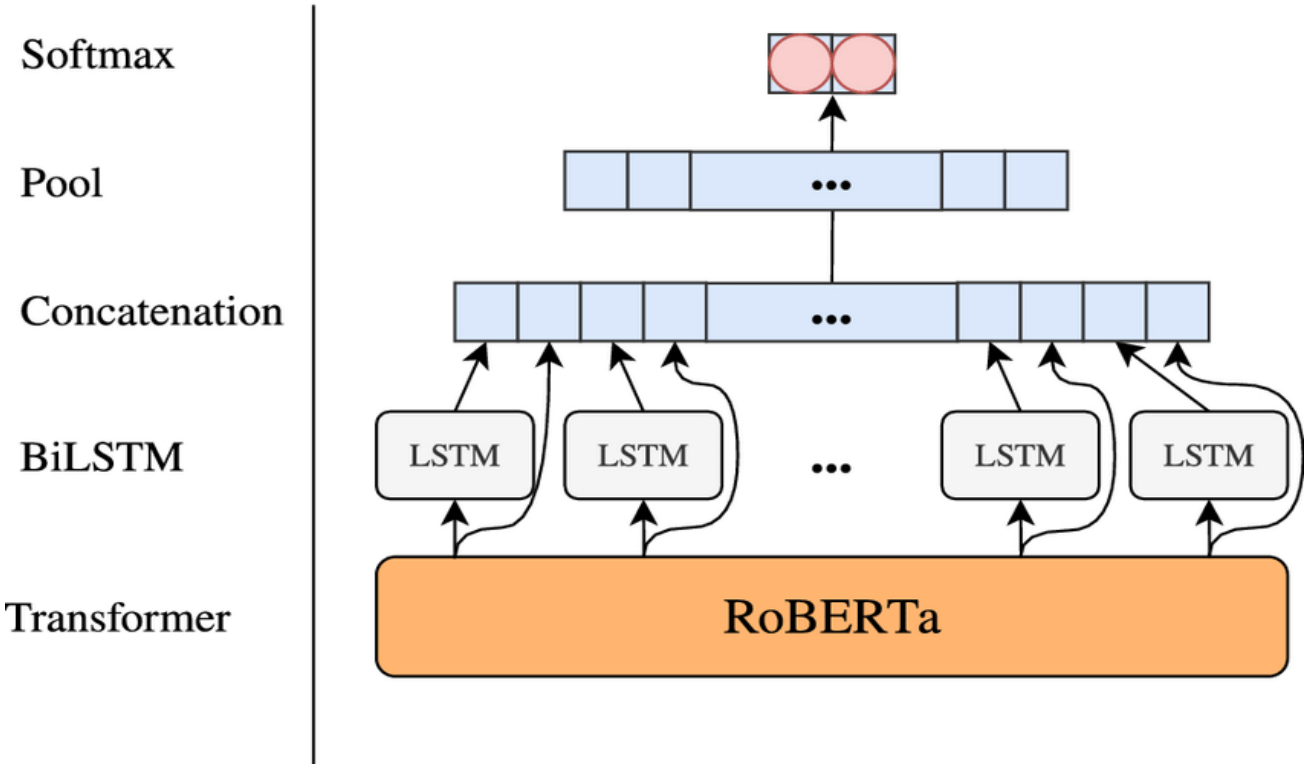
**Figure 5:** Working of ELECTRA framework in the project



- **ROBERTA:** Language model pretraining has led to significant performance gains but the careful comparison between different approaches is challenging. Training is computationally expensive, often done on private datasets of different sizes, and, as we will show, hyperparameter choices have a significant impact on the final results. We present a replication study of BERT pretraining (Devlin et al., 2019) that carefully measures the impact of many key hyperparameters and training data size. We find that BERT was significantly undertrained, and can match or exceed the performance of every model published after it. Our best model achieves state-of-the-art results on GLUE, RACE and SQuAD. These results highlight the importance of previously overlooked design choices and raise questions about the source of recently reported improvements. We release our models and code.



**Figure 6:** Working of ROBERTA framework in the project



## 8. Deep Learning

Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge. ... While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.

The adjective "deep" in deep learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, but that a network with a nonpolynomial activation function with one hidden layer of unbounded width can. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part.

## 9. History of Deep Learning

Deep Learning, as a branch of Machine Learning, employs algorithms to process data and imitate the thinking process, or to develop *abstractions*. Deep Learning (DL) uses layers of algorithms to process data, understand human speech, and visually recognize objects. Information is passed through each layer, with the output of the previous layer providing input for the next layer. The first layer in a network is called the input layer, while the last is called an output layer. All the layers between the two are referred to as hidden layers. Each layer is typically a simple, uniform algorithm containing one kind of activation function.

Feature extraction is another aspect of Deep Learning. Feature extraction uses an algorithm to automatically construct meaningful "features" of the data for purposes of training, learning, and understanding. Normally the Data Scientist, or programmer, is responsible for feature extraction.



The history of Deep Learning can be traced back to 1943, when Walter Pitts and Warren McCulloch created a computer model based on the neural networks of the human brain. They used a combination of algorithms and mathematics they called "threshold logic" to mimic the thought process. Since that time, Deep Learning has evolved steadily, with only two significant breaks in its development. Both were tied to the infamous Artificial Intelligence winters.

Henry J. Kelley is given credit for developing the basics of a continuous Back Propagation Model in 1960. In 1962, a simpler version based only on the chain rule was developed by Stuart Dreyfus. While the concept of back propagation (the backward propagation of errors for purposes of training) did exist in the early 1960s, it was clumsy and inefficient, and would not become useful until 1985.

The earliest efforts in developing Deep Learning algorithms came from Alexey Grigoryevich Ivakhnenko (developed the *Group Method of Data Handling*) and Valentin Grigor′evich Lapa (author of *Cybernetics and Forecasting Techniques)* in 1965. They used models with polynomial (complicated equations) activation functions, that were then analyzed statistically. From each layer, the best statistically chosen features were then forwarded on to the next layer (a slow, manual process).

During the 1970's the first AI winter kicked in, the result of promises that couldn't be kept. The impact of this lack of funding limited both DL and AI research. Fortunately, there were individuals who carried on the research without funding.

The first “convolutional neural networks” were used by Kunihiko Fukushima. Fukushima designed neural networks with multiple pooling and convolutional layers. In 1979, he developed an artificial neural network, called Neocognitron, which used a hierarchical, multi-layered design. This design allowed the computer the “learn” to recognize visual patterns. The networks resembled modern versions, but were trained with a reinforcement strategy of recurring activation in multiple layers, which gained strength over time. Additionally, Fukushima’s design allowed important features to be adjusted manually by increasing the “weight” of certain connections.

Many of the concepts of Neocognitron continue to be used. The use of top-down connections and new learning methods have allowed for a variety of neural networks to be realized. When more than one pattern is presented at the same time, the Selective Attention Model can separate and recognize individual patterns by shifting its attention from one to the other. (The same process many of us use when multitasking). A modern Neocognitron can not only identify patterns with missing information (for example, an incomplete number 5), but can also complete the image by adding the missing information. This could be described as “inference.”

Back propagation, the use of errors in training Deep Learning models, evolved significantly in 1970. This was when Seppo Linnainmaa wrote his master’s thesis, including a FORTRAN code for back propagation. Unfortunately, the concept was not applied to neural networks until 1985. This was when Rumelhart, Williams, and Hinton demonstrated back propagation in a neural network could provide “interesting” distribution representations. Philosophically, this discovery brought to light the question within cognitive psychology of whether human understanding relies on symbolic logic (computationalism) or distributed representations (connectionism). In 1989, Yann LeCun provided the first practical demonstration of backpropagation at Bell Labs. He combined convolutional neural networks with back propagation onto read

"handwritten" digits. This system was eventually used to read the numbers of handwritten checks.

This time is also when the second AI winter (1985-90s) kicked in, which also effected research for neural networks and Deep Learning. Various overly-optimistic individuals had exaggerated the "immediate" potential of Artificial Intelligence, breaking expectations and angering investors. The anger was so intense, the phrase Artificial Intelligence reached pseudoscience status. Fortunately, some people continued to work on AI and DL, and some significant advances were made. In 1995, Dana Cortes and Vladimir Vapnik developed the support vector machine (a system for mapping and recognizing similar data). LSTM (long short-term memory) for recurrent neural networks was developed in 1997, by Sepp Hochreiter and Juergen Schmidhuber.

The next significant evolutionary step for Deep Learning took place in 1999, when computers started becoming faster at processing data and GPU (graphics processing units) were developed. Faster processing, with GPUs processing pictures, increased computational speeds by 1000 times over a 10-year span. During this time, neural networks began to compete with support vector machines. While a neural network could be slow compared to a support vector machine, neural networks offered better results using the same data. Neural networks also have the advantage of continuing to improve as more training data is added.

Around the year 2000, *The Vanishing Gradient Problem* appeared. It was discovered "features" (lessons) formed in lower layers were not being learned by the upper layers, because no learning signal reached these layers. This was not a fundamental problem for all neural networks, just the ones with gradient-based learning methods. The source of the problem turned out to be certain activation functions. A number of activation functions condensed their input, in turn reducing the output range in a somewhat chaotic fashion. This produced large areas of input mapped over an extremely small range. In these areas of input, a large change will

be reduced to a small change in the output, resulting in a vanishing gradient. Two solutions used to solve this problem were layer-by-layer pre-training and the development of long short-term memory.



In 2001, a research report by META Group (now called Gartner) described he challenges and opportunities of data growth as three-dimensional. The report described the increasing volume of data and the increasing speed of data as increasing the range of data sources and types. This was a call to prepare for the onslaught of Big Data, which was just starting.

In 2009, Fei-Fei Li, an AI professor at Stanford launched ImageNet, assembled a free database of more than 14 million labelled images. The Internet is, and was, full of unlabelled images. Labelled images were needed to "train" neural nets. Professor Li said, "Our vision was that Big Data would change the way machine learning works. Data drives learning."

By 2011, the speed of GPUs had increased significantly, making it possible to train convolutional neural networks "without" the layer-by-layer pre-training. With the increased computing speed, it became obvious Deep Learning had significant advantages in terms of efficiency and speed. One example is AlexNet, a convolutional neural network whose architecture won several international competitions during 2011 and 2012. Rectified linear units were used to enhance the speed and dropout.

Also, in 2012, Google Brain released the results of an unusual project known as *The Cat Experiment*. The free-spirited project explored the difficulties of "unsupervised learning." Deep Learning uses "supervised learning," meaning the convolutional neural net is trained using labelled data (think images from ImageNet). Using unsupervised learning, a convolutional neural net is given unlabelled data, and is then asked to seek out recurring patterns.

The Cat Experiment used a neural net spread over 1,000 computers. Ten million "unlabelled" images were taken randomly from YouTube, shown to the system, and then the training software

was allowed to run. At the end of the training, one neuron in the highest layer was found to respond strongly to the images of cats. Andrew Ng, the project's founder said, "We also found a neuron that responded very strongly to human faces." Unsupervised learning remains a significant goal in the field of Deep Learning.



The Cat Experiment works about 70% better than its forerunners in processing unlabelled images. However, it recognized less than a 16% of the objects used for training, and did even worse with objects that were rotated or moved.

Currently, the processing of Big Data and the evolution of Artificial Intelligence are both dependent on Deep Learning. Deep Learning is still evolving and in need of creative ideas.

## 10. Tasks performed through Deep Learning

Deep learning applications are used in industries from automated driving to medical devices.

**Automated Driving:** Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

**Aerospace and Defense:** Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.

**Medical Research:** Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.

**Industrial Automation:** Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.

**Electronics:** Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

# External Interface Requirements:



## 1. User Interface:

- It will be quick in responding and user-friendly.
- User can enter the input and generate the results.

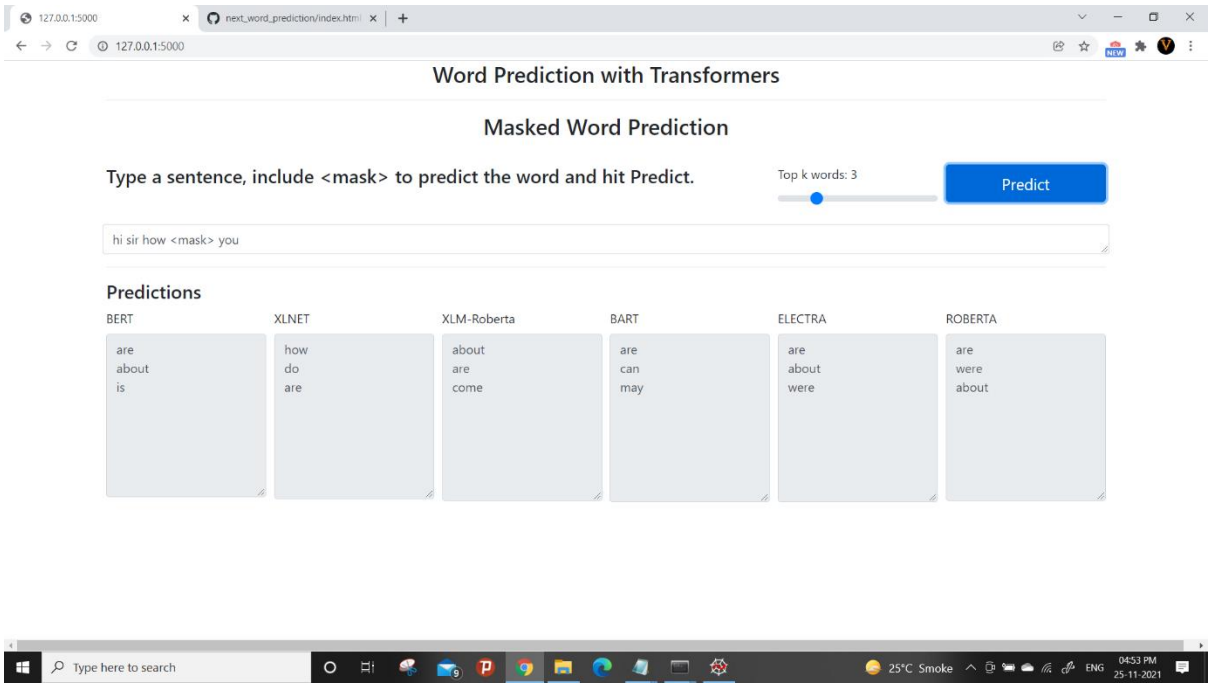
## 2. Software Interface:

This whole project works on browsers like Chrome, Firefox etc. and is based on the technologies like HTML, CSS and JavaScript.

## 3. Communication Interface:

There will be multiple suggestions for the user to choose from once the prediction is completed.

## Frontend Part of the project:



## Other Non-Functional Requirements:



1. **Performance Requirements**
   It gives quick responses. It is time saving.

2. **Portability Requirements**
   User can use it from anywhere.

3. **Availability Requirements**
   This system is up and running whenever needed.

4. **Scalability Requirements**
   It meets the needs for which it was build.

5. **Usability Requirements**
   The user will be able to find any missing words easily offline.

6. **Supportability Requirements**
   It is supported by all the browsers like Chrome, Firefox etc.

7. **Efficiency Requirements**
   It works with good efficiency.

# CONCLUSION AND FUTURE SCOPE



**CONCLUSION:** This application will be very beneficial to people as they do not have to search for different words separately while keeping the context of sentences in mind. They can use it for instant solution with decent accuracy.

**FUTURE SCOPE:** I think this project will be very beneficial for students and professionals who struggle with vocabulary and grammar as it will surely save their time. It will also improve the quality of work done by any user since it'll be without any major errors.

## REFERENCES:

[1] https://towardsdatascience.com/

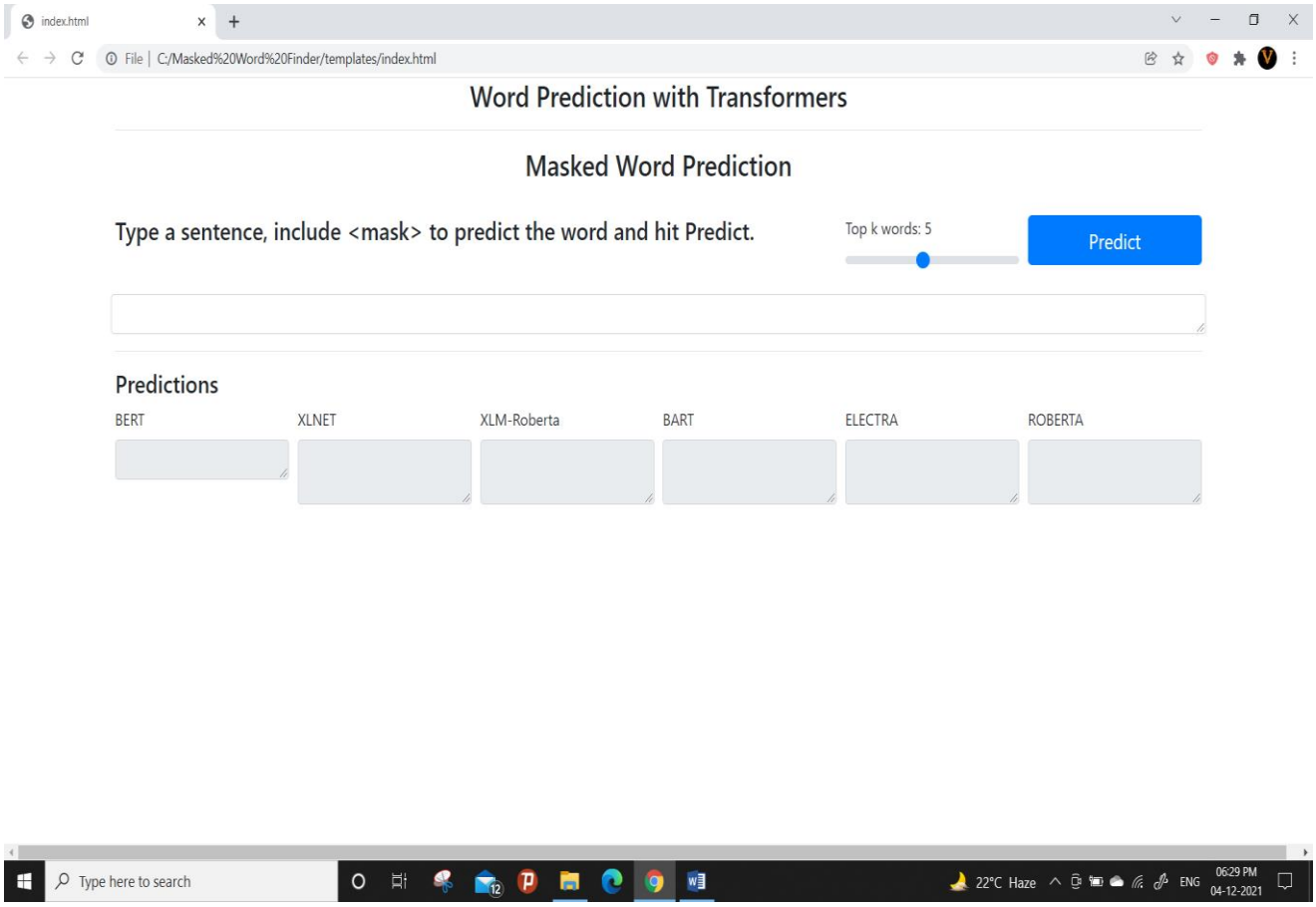
[2] https://stackoverflow.com/

[3] https://www.dataversity.net/

# SNAPSHOTS:

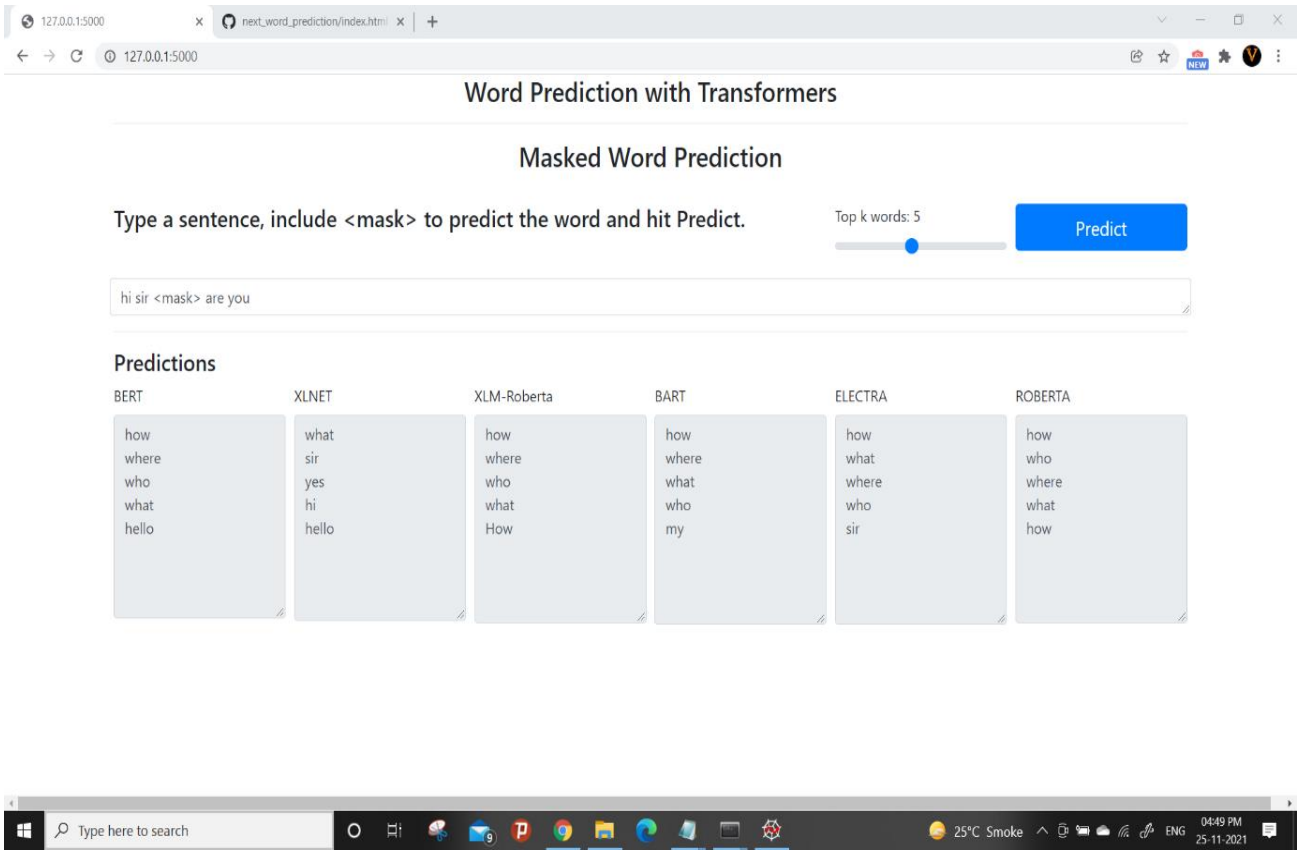
## 1) Frontend before giving the input





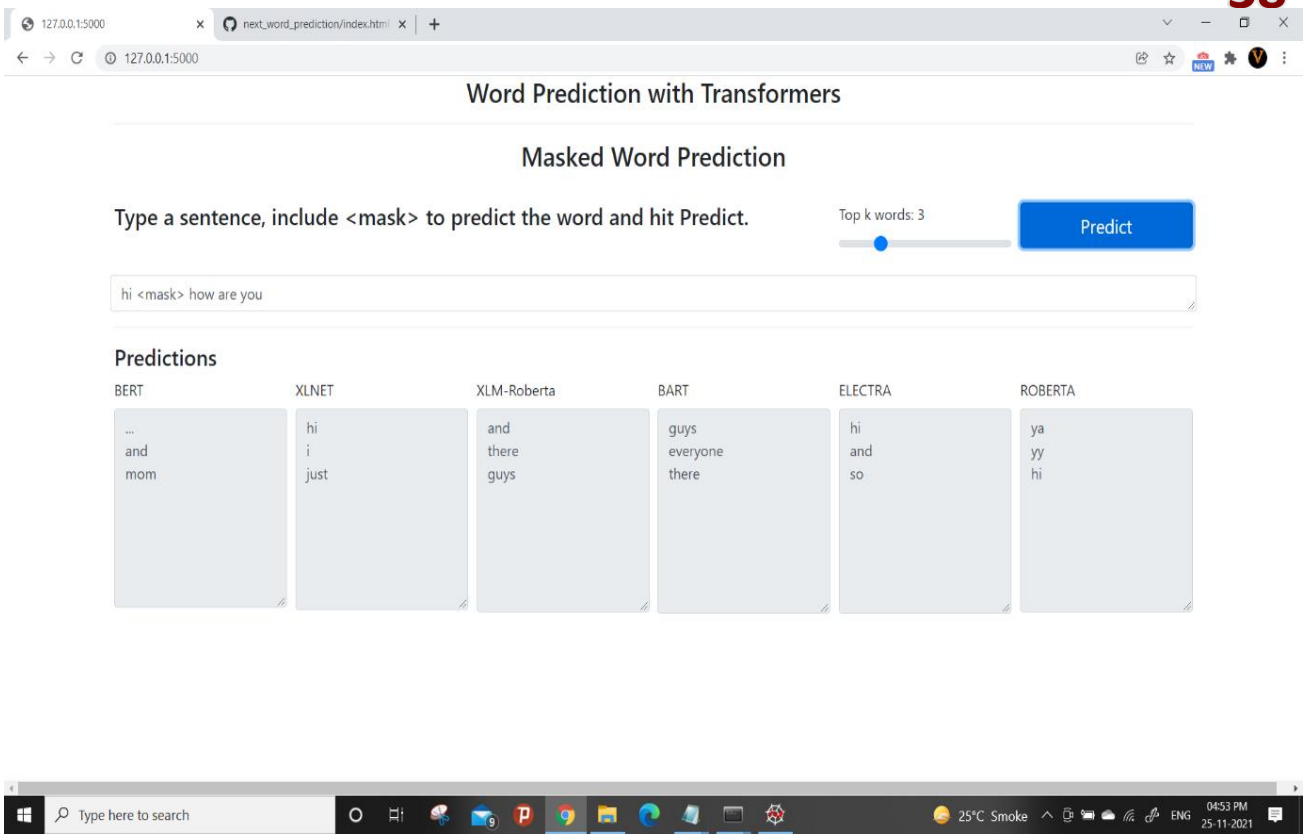
# 2) Word Prediction through different frameworks at the same time





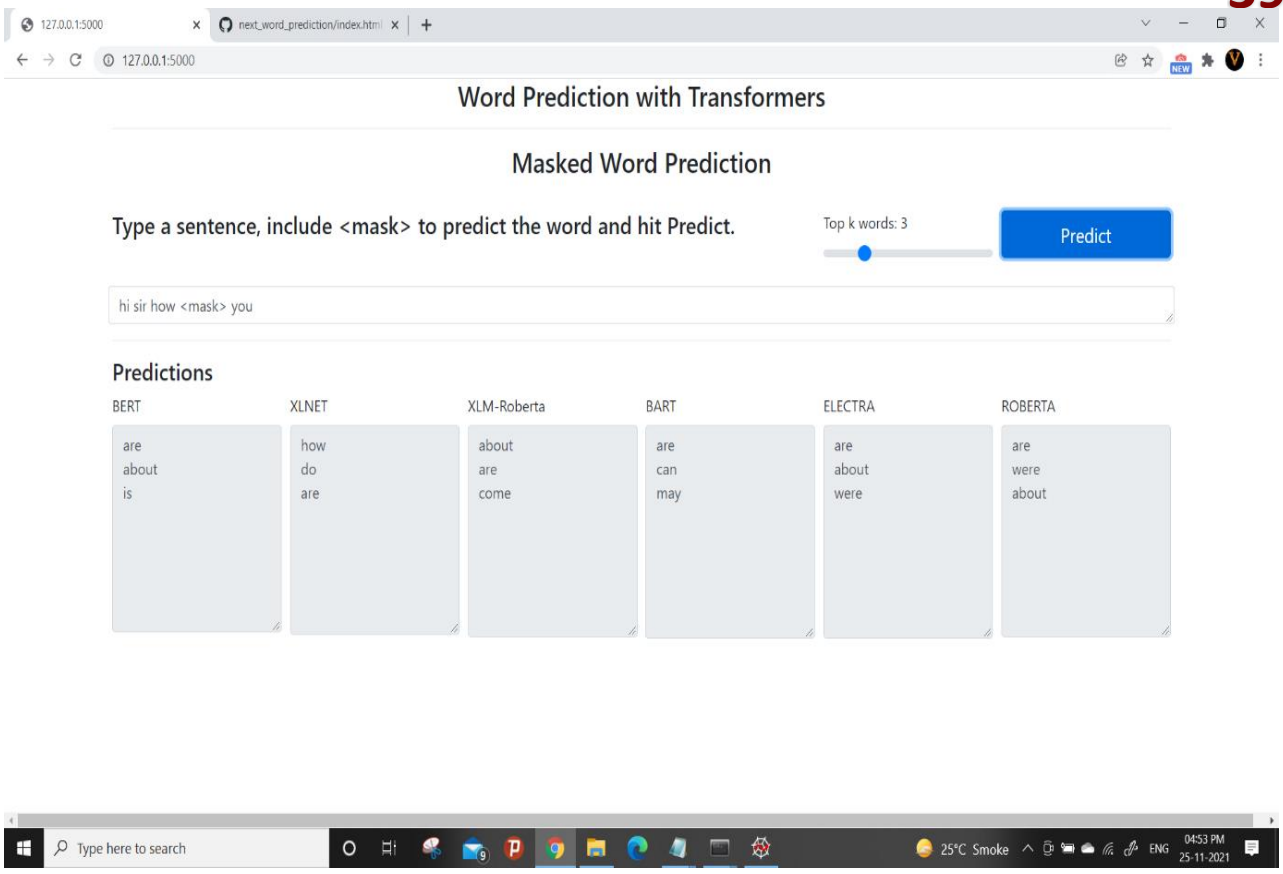
# 3) Changing number of required works using the scroller





# 4) Predicting different words in same sentence to ensure the accuracy of prediction





# Points to avoid errors while running the project and using it correctly:



1. Keep all files in the same folder.

2. Make sure Spyder or any other Python IDE is downloaded in your system.

3. Don’t forget to download all the required libraries (torch, transformer etc.) to run the project or else you’ll face errors/warnings while running it.

4. Go through the How To Run text file and follow the instructions to run the project smoothly.

5. Wait for the complete execution of one file before running the other.

6. Don’t use Ctrl + C to copy URL from command prompt because it will end the execution of app.py. Type it on the browser manually.