CareerNinja is building a personalized &amp; experiential learning platform to make high-quality learning accessible at a mass global scale. It is revolutionizing the way youth kickstart & upgrade their career. At the core of it is a unique adaptive learning product with an exclusive technology that enables course creation with 1/10th the effort & cost.

**Machine Learning Engineer**  
We are looking for an expert in Data Engineering Machine Learning, Deep Learning, and NLP. You will lead all the processes from R&D, data collection, cleaning, and pre-processing, to training models and deploying them to production. The ideal candidate will be passionate about artificial intelligence and stay up-to-date with the latest developments in the field. Most of our work revolves around text data so we need a candidate who has proven expertise in recent advancements in the field of NLP.  
  
**Responsibilities:**  
\* Data Collection and Curation.  
\* Select appropriate annotated datasets for Supervised Learning methods.  
\* Use effective text representations to transform natural language into useful features.  
\* Find and implement the right algorithms and tools for NLP tasks.  
\* Develop ML/DL/NLP models according to requirements.  
\* Train the developed model and run evaluation experiments.  
\* Perform statistical analysis of results and refine models.  
\* Extend ML libraries and frameworks to apply in NLP tasks.  
\* Building recommendation engines to increase the personalization of content for the user.  
\* Testing and deployment of models on cloud services i.e. AWS/GCP etc.  
\* Act as a bridge between product and engineering team to translate product requirements into technical solutions.  
\* Stay updated and use industry best practices and latest technology trends.  
\* Prototype ideas rapidly using cutting-edge technologies, drive architecture discussions, and propose solutions to system and product changes.  
  
**Assignment:  Scenario**  
We are building a system that can **extract topics** from an article and **ranking of the documents** according to relevance to the query.  We want to build a system that can give us the most accurate results for the above use cases for different domain-specific articles. As per our research, it is proved that recent techniques developed in the field of NLP give exceptional SOTA results for such use cases i.e. Transformer models (BERT, GPT-1/2/3, etc.) and Transfer Learning.  
  
**Expectations**  
We need to build a domain-specific keyword/topic extraction system that can extract the most relevant/important topics from the text. For example, we have a query then we need to extract such keywords/topics/tags which define the syllabus or subqueries of the main query.  For eg. the main query is Search Engine Optimization then its output must contain its subtopics extracted from the articles eg. “On-page SEO, Off-page SEO, Technical SEO” or other relevant keywords which define the query. And then be able to break those down into the second level of the topic tree, ie. A cluster of sub-sub-topics. Again we need such a solution to this problem which can give us results with the same accuracy with all domains i.e. Technology, Science, Engineering, Marketing, Management, Economics, History, etc. Similarly, for the other use cases, we need a model or different models which can rank the articles fetched from the internet for a given query.  Input will be a query and you need to fetch articles from the internet for that query and perform the above use cases and produce the required output. Required output for a query

1.    Top 5 ranked articles from the fetched articles for that query

2.    Keywords/Topics/Tags from that article

For eg. the query is “How to do SEO” then the output should contain the top 5 articles from the internet which are most relevant and cover most of the content, topics/keywords from each article.

Kindly provide us with the following:

* **Approach to the solution:**
  1. Which ML/DL model architecture you used and why?
  2. How would you ensure the scalability of your solution?
  3. Is there a need for any dataset? If yes then how much data is sufficient to train the model in order to get the required results?
  4. Is there a need to create manual datasets, if yes then what parameters and sample size did you consider to create a dataset?
  5. Are your model and dataset generalized enough for different domains of the use cases, How?
  6. How would you train, test and deploy your model to production?
  7. How would you perform hyperparameter tuning on your model to improve accuracy?
  8. Anything else you want to let us know about your approach.
* Model/Code on GitHub repo or Colab Notebook with the necessary documentation describing the model functioning.
* Only Approach to **generate different types of questions** (short answer type, MCQs, true/false, fill in the blanks, long answer type, etc.) for that same article.