**Questions with MCQs Generation from Text using T5 and NLP**

**Overview:**

This code implements a system for generating multiple-choice questions (MCQs) from a given input text using techniques from natural language processing (NLP) and machine learning. It uses pre-trained models such as T5 (Text-To-Text Transfer Transformer) and Sense2Vec along with various NLP techniques to extract relevant information and formulate questions based on the input text.

**Components**:

**Sense2Vec Model:** This model is used to obtain word embeddings and identify the best sense of a word within the context of the text. It helps in determining the appropriate answer choices for the generated questions.

**T5 Model:** The T5 model is a transformer-based architecture that is fine-tuned on the SQuAD dataset. It is employed for generating questions from the input text using the "context: ..." prefix.

**NLP Techniques:**

**Tokenization:** The text is tokenized into sentences using the sent\_tokenize function from the NLTK library.

**Keyword Extraction**: Nouns and noun phrases are extracted from the text using the MultipartiteRank algorithm to identify potential keywords for generating questions.

**Sentence Filtering:** Sentences containing the identified keywords are filtered to ensure relevance to the generated questions.

**Question Generation:**

The T5 model generates questions based on the input text and identified keywords. Each question is prefixed with "context: ..." and suffixed with "answer: {keyword}".

The generated questions are then processed to obtain answer choices using Sense2Vec. Distractors or incorrect answer choices are chosen based on semantic similarity to the correct answer.

**Code Breakdown:**

**MCQs\_available Function:** Determines if a word has a corresponding sense available in the Sense2Vec model.

**Edits Function:** Generates all possible edits (deletions, transpositions, replacements, and insertions) for a given word.

**Sense2Vec\_get\_words Function:** Retrieves words similar to a given word using the Sense2Vec model.

**Get\_options Function:** Obtains distractors or answer choices for a given correct answer using Sense2Vec.

**Tokenize\_sentences Function:** Tokenizes the input text into sentences using NLTK's sent\_tokenize.

**Get\_sentences\_for\_keyword Function:** Extracts sentences containing keywords from the input text.

**Is\_far Function:** Determines if a word is significantly different from a list of words based on the normalized Levenshtein distance.

**Filter\_phrases Function:** Filters out phrases that are too similar to each other based on the normalized Levenshtein distance.

**Get\_nouns\_multipartite Function**: Extracts nouns and noun phrases from the input text using the MultipartiteRank algorithm.

**Get\_phrases Function**: Extracts noun phrases from the input text.

**Get\_keywords Function:** Obtains keywords from the input text based on extracted nouns and noun phrases.

**Generate\_questions Function:** Utilizes the T5 model to generate questions and Sense2Vec to generate answer choices.

**PythonPredictor Class:** Main class responsible for integrating all components and orchestrating the MCQ generation process.

**Predict Function:** Entry point for generating MCQs from the input text. It orchestrates the entire process, from keyword extraction to question generation.

**Usage:**

To use the system, provide the input text to the predict function of the PythonPredictor class along with the maximum number of questions to generate. The function returns a JSON object containing the input text, generated questions, and the time taken for the process.

**Dependencies:**

**Sense2Vec**: Pre-trained model for obtaining word embeddings and sense disambiguation.

**T5** **Model**: Pre-trained model for text generation tasks.

**NLTK**: Natural Language Toolkit for tokenization and sentence processing.

**spaCy**: NLP library for advanced text processing tasks.

**PyTorch**: Deep learning library used for neural network computations.

**Transformers**: Library for working with transformer-based models like T5.

**Gensim**: Library for topic modeling, document indexing, and similarity retrieval.

**Frequency** **Distributions** (**FreqDist**): Utility from NLTK for calculating word frequencies.

**Levenshtein** **Distance**: Library for calculating string similarity.

**Conclusion**:

The MCQ generation system presented in this documentation offers a robust solution for automatically creating multiple-choice questions from textual content. By leveraging state-of-the-art NLP techniques and pre-trained models, it streamlines the process of formulating questions, thereby saving time and effort for educators, content creators, and anyone involved in generating assessments or quizzes.

This documentation provides a comprehensive overview of the code's functionality, explaining each component's purpose and usage. It aims to facilitate understanding and usage for developers or users interested in implementing or utilizing the MCQ generation system.