1. Introduction to NLP

Video 1: Introduction to NLP

- **Definition**: NLP enables machines to understand and process human language.
- **Applications**: Sentiment analysis, spam detection, translation, chatbots.
- **Challenges**: Ambiguity, context, multiple meanings of words.

2. Text Preprocessing

Videos 2–5: Tokenization, Stemming, Lemmatization, Stopword Removal

- **Tokenization**: Splitting text into words or sentences.
  `nltk.word_tokenize("Hello world!") → ['Hello', 'world', '!']`
- **Stemming**: Truncates words to their root forms.
  `stemming("studying") → "study"`
- **Lemmatization**: Converts words to their dictionary base forms using vocabulary.
  `lemmatization("better") → "good"`
- **Stopword removal**: Common words like “the”, “is” are removed to reduce noise.
  Done using `nltk.corpus.stopwords`.

3. Text Representation Techniques

Videos 6–9: BoW, TF-IDF, Word2Vec

- **Bag of Words**: Simple word frequency representation; loses context.
- **TF-IDF**: Reduces weight for common words, boosts rare but important words.
  TF=count of termtotal terms,IDF=log(Ndf(t))TF = \frac{\text{count of term}}{\text{total terms}}, \quad IDF = \log\left(\frac{N}{df(t)}\right)TF=total termscount of term,IDF=log(df(t)N)
- **Word2Vec (CBOW/Skip-gram)**:
  - CBOW: Predict target word from context.
  - Skip-gram: Predict context from target word.
  - Learns semantic similarity: `vector("king") - vector("man") + vector("woman") ≈ vector("queen")`

4. Part-of-Speech Tagging & Named Entity Recognition

Videos 10–11: POS Tagging, NER

- **POS tagging**: Identifies grammatical parts like noun, verb, adjective.
  - Uses `nltk.pos_tag()` or spaCy models.

- **NER**: Detects entities like persons, organizations, locations.
    - With spaCy: `ent.label_` → "PERSON", "ORG", "GPE"

5. Text Classification with Machine Learning

Videos 12–14: Naive Bayes, Logistic Regression, Pipeline

- **Naive Bayes**:
    - Based on Bayes' theorem with word independence assumption.
    - Works well for spam detection, sentiment classification.

- **Logistic Regression**:
    - Uses sigmoid to classify into binary/multiclass.
    - Input is usually TF-IDF or BoW vectors.

- **Pipeline**:
    - Automates steps: preprocessing → vectorizing → model training.

6. Advanced ML Algorithms for NLP

Videos 15–16: SVM, Cross-Validation

- **SVM**: Finds hyperplane that best separates text classes.
- **Cross-validation**: Evaluates model performance reliably.

7. Deep Learning Models in NLP

Videos 17–20: RNN, LSTM, GRU

- **RNN**:
    - Processes sequence data.
    - Suffers from vanishing gradients.

- **LSTM**:

    - Introduces memory cells and gates (forget, input, output).
    - Helps with long-term dependencies in text.

- **GRU**:
    - Simpler than LSTM, but performs similarly in many tasks.

8. Attention Mechanism & Transformer

Videos 21–22: Attention, Transformer Architecture

- **Attention**:
    - Weights input tokens based on relevance.
    - Computes context vectors for each output.

- **Transformer**:

    - Encoder-decoder with multi-head self-attention and feedforward layers.
    - Enables massive parallelism (no recurrence like RNNs).
    - Forms the basis for models like BERT, GPT.

9. BERT & Hugging Face

Videos 23–24: BERT Usage & Hugging Face

- **BERT**:
    - Pretrained using Masked Language Modeling (MLM).
    - Outputs contextual embeddings.

- **Hugging Face Transformers**:

    - `pipeline('sentiment-analysis')` or `AutoModelForSequenceClassification`.

    - Easy to fine-tune on custom data using `Trainer` API.

10. Projects & Deployment

Videos 25–28: End-to-End Projects

- **Text Classification Project**:
    - Load data → clean → vectorize (TF-IDF) → train model → evaluate.
    - Export via `pickle` or `joblib`.

- **Chatbot using Flask**:
    - Uses cosine similarity on TF-IDF or Hugging Face models.
    - Flask serves a web UI for user interaction.

- **Multi-class Classification**:
    - Adapts logistic regression or deep models for >2 categories.

- **Model Deployment**:
    - Hosting model via Flask + HTML template for user input.