টেক্সট প্রি-প্রসেসিং হলো এনএলপি পাইপলাইনের একটি অত্যন্ত গুরুত্বপূর্ণ ধাপ, যা ডেটা সংগ্রহের পরে এবং ফিচার ইঞ্জিনিয়ারিংয়ের আগে সম্পন্ন করা হয়। এর মূল লক্ষ্য হলো অগোছালো টেক্সট ডেটাকে পরিষ্কার করে মডেলের জন্য উপযোগী করে তোলা। নিচে এই প্রক্রিয়ার প্রতিটি ধাপ বিস্তারিতভাবে ব্যাখ্যা করা হলো:

১. লোয়ারকেসিং (Lowercasing)

টেক্সট প্রি-প্রসেসিংয়ের প্রাথমিক ধাপ হলো পুরো টেক্সটকে ছোট হাতের অক্ষরে রূপান্তর করা।

- **কেন প্রয়োজন:** পাইথন একটি কেস-সেনসিটিভ প্রোগ্রামিং ল্যাঙ্গুয়েজ। ফলে মডেলের কাছে "Basically" এবং "basically" দুটি ভিন্ন শব্দ হিসেবে গণ্য হতে পারে। এটি মডেলের মধ্যে অপ্রয়োজনীয় জটিলতা (complexity) তৈরি করে। সব শব্দকে লোয়ারকেস করলে মডেল এই দুটিকে একই শব্দ হিসেবে চিনতে পারে।
- **প্রয়োগ:** পাইথনের `lower()` ফাংশন ব্যবহার করে একটি স্ট্রিং বা পুরো ডেটাসেটের কলামকে লোয়ারকেস করা যায়।

২. অপ্রয়োজনীয় এলিমেন্ট রিমুভাল (HTML Tags & URL Removal)

যখন ডেটা ওয়েবসাইট থেকে স্ক্র্যাপ করে আনা হয়, তখন তাতে অনেক অপ্রয়োজনীয় এলিমেন্ট থাকে।

- **HTML Tags:** ওয়েবসাইট থেকে ডেটা নিলে তাতে `<br>`, `<div>`, বা `<a>` এর মতো ট্যাগ থাকতে পারে। এগুলো ব্রাউজারের জন্য প্রয়োজনীয় হলেও সেন্টিমেন্ট অ্যানালাইসিস বা এনএলপি মডেলের জন্য কোনো অর্থ বহন করে না।
- **URLs:** সোশ্যাল মিডিয়া বা চ্যাট ডেটাতে অনেক লিঙ্ক বা ইউআরএল থাকে। ক্লাসিফিকেশন বা মডেলিংয়ের ক্ষেত্রে এগুলো সাধারণত কোনো ভূমিকা রাখে না এবং মডেলকে কনফিউজড করতে পারে।
- **পদ্ধতি:** এই ট্যাগ এবং ইউআরএল সরানোর জন্য **Regular Expression (Regex)** ব্যবহার করা হয়।

৩. পাংচুয়েশন রিমুভাল (Removing Punctuations)

ইংরেজি ভাষার বিভিন্ন বিরামচিহ্ন যেমন— ফুলস্টপ (.), কমা (,), প্রশ্নবোধক চিহ্ন (?) ইত্যাদি সরিয়ে ফেলা হয়।

- **কেন প্রয়োজন:** পাংচুয়েশন থাকলে টোকেনাইজেশনের সময় "Hello!" এবং "Hello" দুটি আলাদা শব্দ বা টোকেন হিসেবে ধরা হয়। এতে মডেলের শব্দভাণ্ডার (vocabulary) অকারণে বড় হয়ে যায়।
- **প্রয়োগ:** পাইথনের `string.punctuation` লাইব্রেরি ব্যবহার করে খুব দ্রুত সব পাংচুয়েশন রিমুভ করা সম্ভব। বড় ডেটাসেটের ক্ষেত্রে `str.translate` পদ্ধতিটি সবচেয়ে দ্রুত কাজ করে।

৪. চ্যাট ওয়ার্ড ট্রিটমেন্ট (Chat Word Treatment)

আজকাল মেসেজিং অ্যাপে আমরা অনেক সংক্ষিপ্ত রূপ বা শর্টহ্যান্ড ব্যবহার করি (যেমন: ASAP, LOL, GN, ROFL)।

- **কেন প্রয়োজন:** মডেলকে সঠিক অর্থ বোঝানোর জন্য এই সংক্ষিপ্ত শব্দগুলোকে তাদের পূর্ণরূপে রূপান্তর করা প্রয়োজন। যেমন- "GN" কে "Good Night" এ রূপান্তর করা।
- **পদ্ধতি:** এটি করার জন্য সাধারণত একটি ডিকশনারি ব্যবহার করা হয় যেখানে সংক্ষিপ্ত রূপ এবং তাদের পূর্ণরূপের ম্যাপিং থাকে।

৫. স্পেলিং কারেকশন (Spelling Correction)

টাইপিংয়ের সময় হওয়া ভুল বানান সংশোধন করা একটি গুরুত্বপূর্ণ ধাপ।

- **কেন প্রয়োজন:** ভুল বানানের কারণে একই শব্দের একাধিক রূপ তৈরি হতে পারে যা মডেলের পারফরম্যান্স কমিয়ে দেয়।
- **প্রয়োগ:** পাইথনের **TextBlob** লাইব্রেরি ব্যবহার করে খুব সহজে সাধারণ বানানের ভুলগুলো ঠিক করা যায়। তবে রিজিওনাল বা বিশেষ ধরনের ডেটার ক্ষেত্রে কাস্টম স্পেল চেকার প্রয়োজন হতে পারে।

৬. স্টপ-ওয়ার্ডস রিমুভাল (Removing Stopwords)

ভাষার সেই সাধারণ শব্দগুলো (যেমন: is, the, and, in, of) যা বাক্য গঠনে সাহায্য করে কিন্তু বাক্যের মূল ভাবে বিশেষ কোনো গুরুত্ব রাখে না, সেগুলোকে স্টপ-ওয়ার্ডস বলে।

- **কেন প্রয়োজন:** এগুলো সরিয়ে ফেললে ডেটার সাইজ কমে এবং মডেলটি কেবল গুরুত্বপূর্ণ শব্দগুলোর ওপর মনোযোগ দিতে পারে। তবে পার্টস অফ স্পিচ (POS) ট্যাগিংয়ের মতো কাজে এগুলো সরানো হয় না।
- **প্রয়োগ:** **NLTK** লাইব্রেরিতে ইংরেজি-সহ বিভিন্ন ভাষার স্টপ-ওয়ার্ডসের লিস্ট দেওয়া আছে।

৭. ইমোজি হ্যান্ডলিং (Handling Emojis)

ইমোজি আমাদের আবেগ প্রকাশের শক্তিশালী মাধ্যম, কিন্তু মেশিন লার্নিং অ্যালগরিদম এগুলো সরাসরি বুঝতে পারে না।

- **সমাধান:** হয় ইমোজিগুলো পুরোপুরি রিমুভ করে দেওয়া যায়, অথবা সেগুলোকে সেগুলোর অর্থ দিয়ে রিপ্লেস করা যায় (যেমন: বদলে 'Happy')। এর জন্য `emoji` লাইব্রেরির `demojize` ফাংশন ব্যবহার করা হয়।

৮. টোকেনাইজেশন (Tokenization)

একটি বড় টেক্সটকে ছোট ছোট অংশে বা 'টোকেন'-এ ভাগ করার প্রক্রিয়াকে টোকেনাইজেশন বলে। এটি শব্দ (Word) বা বাক্য (Sentence) উভয় লেভেলে হতে পারে।

- **চ্যালেঞ্জ:** ইমেইল আইডি, কারেন্সি সিম্বল (যেমন: $20), বা ডট যুক্ত শব্দ (যেমন: U.S.A.) সঠিকভাবে টোকেনাইজ করা বেশ চ্যালেঞ্জিং।
- **পদ্ধতি:** পাইথনের ডিফল্ট `split()` ফাংশন দিয়ে প্রাথমিক কাজ চললেও, উন্নত কাজের জন্য **NLTK** বা **spaCy** লাইব্রেরি ব্যবহার করা হয়। বিশেষ করে spaCy জটিল টেক্সটও খুব নির্ভুলভাবে টোকেনাইজ করতে পারে।

৯. স্টেমিং বনাম লেমাটাইজেশন (Stemming vs Lemmatization)

এই দুটি পদ্ধতির কাজই হলো একটি শব্দকে তার মূল রূপে (Root Word) ফিরিয়ে আনা।

- **স্টেমিং (Stemming):** এটি একটি রুল-বেসড পদ্ধতি যা শব্দের শেষের অংশ কেটে দেয়। এর ফলে পাওয়া মূল শব্দটি সবসময় অর্থপূর্ণ নাও হতে পারে (যেমন: 'probably' হয়ে যায় 'probl')। এটি খুব দ্রুত কাজ করে।
- **লেমাটাইজেশন (Lemmatization):** এটি একটি ডিকশনারি বা 'WordNet' ভিত্তিক পদ্ধতি যা শব্দের ব্যাকরণগত মূল রূপ বের করে যা সবসময় অর্থপূর্ণ হয় (যেমন: 'went' হয়ে যায় 'go')। এটি স্টেমিংয়ের চেয়ে কিছুটা ধীর কিন্তু অনেক বেশি নির্ভুল।

- **পার্থক্য:** যদি স্পিড দরকার হয় তবে স্টেমিং ব্যবহার করা হয় (যেমন: ইনফরমেসন রিট্রিভাল বা গুগল সার্চ), আর সঠিক অর্থপূর্ণ শব্দ দরকার হলে লেমাটাইজেশন ব্যবহার করা হয়।

অ্যাসাইনমেন্টের প্রেক্ষাপট

উৎস অনুযায়ী, এই থিওরিগুলো শেখার পর **TMDB API** ব্যবহার করে মুভি ডেটাসেট তৈরি করে সেগুলোর ডেসক্রিপশনের ওপর এই প্রি-প্রসেসিং ধাপগুলো প্রয়োগ করার পরামর্শ দেওয়া হয়েছে। এতে বাস্তব জীবনের ডেটা নিয়ে কাজ করার অভিজ্ঞতা তৈরি হয়।

সংক্ষেপে বলতে গেলে, টেক্সট প্রি-প্রসেসিং হলো রান্নার আগে সবজি ধোয়া এবং কাটার মতো; এটি সরাসরি মডেল তৈরি না করলেও একটি ভালো মডেল তৈরির জন্য অপরিহার্য।