Session 01&02

Introduction to NLP
Text Preprocessing

NLP | Machine Learning | Zahra Amini

Telegram: @zahraamini_ai & Instagram:@zahraamini_ai & LinkedIn: @zahraamini-ai

https://zil.ink/zahraamini

معرفی و تعریف پردازش زبان طبیعی (NLP)

NLP چیست؟

پردازش زبان طبیعی، شاخه‌ای از علوم کامپیوتر، هوش مصنوعی و یادگیری ماشین است که به تعامل بین کامپیوترها و زبان انسان (طبیعی) می‌پردازد. هدف اصلیNLP این است که کامپیوترها بتوانند زبان طبیعی را به همان روشی که انسان‌ها درک و تولید می‌کنند، بفهمند و پردازش کنند

- در این زمینه، کامپیوترها با کمک الگوریتم‌های خاص، قادر به تحلیل و تفسیر داده‌های متنی و صوتی به زبان انسانی می‌شوند

این فناوری کاربردهای گسترده‌ای دارد، از جمله ترجمه خودکار، تحلیل احساسات، تشخیص گفتار و چت‌بات‌ها و ...

نقش یادگیری ماشین در NLP چیست؟

مدل‌های یادگیری ماشین، با استفاده از داده‌های زبانی و متنی، یاد می‌گیرند که الگوهای خاص را در داده‌های زبان انسانی شناسایی کنند

این مدل‌ها با کمک یادگیری عمیق نیز تقویت شده‌اند تا بتوانند عملکردهای پیچیده‌تری همچون ترجمه خودکار و خلاصه‌سازی متون را انجام دهند

- در یادگیری ماشین کلاسیک، به عنوان مثال می‌توان از الگوریتم ماشین بردار پشتیبان برای تحلیل احساسات استفاده کرد. فرض کنید یک مجموعه داده از توییت‌ها داریم که هر کدام برچسبی مانند "مثبت"، "منفی" یا "خنثی" دارند. مدل ماشین بردار پشتیبان با استفاده از ویژگی‌های استخراج‌شده از متن (مانند تعداد کلمات مثبت یا منفی) آموزش می‌بیند تا بتواند توییت‌های جدید را طبقه‌بندی کند. پس از آموزش، مدل می‌تواند تشخیص دهد که آیا یک توییت جدید دارای احساس مثبت، منفی یا خنثی است

همچنین، در تشخیص موجودیت‌های نام‌دار (NER)، الگوریتم‌هایی مانند جنگل تصادفی می‌توانند برای شناسایی اسامی افراد، مکان‌ها یا تاریخ‌ها در متن مورد استفاده قرار گیرند. مثلا در متنی که شامل جمله "علی در تهران زندگی می‌کند"، مدل می‌تواند "علی" را به عنوان یک نام و "تهران" را به عنوان یک مکان شناسایی کند

برای استفاده از الگوریتم‌های یادگیری ماشین کلاسیک نیازی به داده‌های عظیم یا معماری‌های پیچیده شبکه عصبی ندارید و با داده‌های محدود و محاسبات ساده‌تر هم قابل پیاده‌سازی هستند

در تحلیل احساسات یا دسته‌بندی متون، می‌توان با استفاده از تکنیک‌های ساده‌تری مانند BOW (Bag of Words) یا TF-IDFو سپس اعمال الگوریتم‌های کلاسیک مانندNaive Bayes ، نتایج مناسبی به دست آورد

پیش‌پردازش متن

پیش‌پردازش متن چیست؟

پیش‌پردازش متن فرآیند آماده‌سازی داده‌های متنی برای ورود به مدل‌های یادگیری ماشین است. داده‌های متنی خام معمولاً دارای نویزهای زیادی هستند که می‌تواند عملکرد مدل‌ها را تضعیف کند. پیش‌پردازش متن به ما کمک می‌کند تا این داده‌ها را به شکلی ساختاریافته و بهینه برای استفاده در مدل‌های یادگیری ماشین تبدیل کنیم

مراحل مختلف پیش‌پردازش متن

❖ حذف نویز Noise Removal

اولین مرحله در پیش‌پردازش، حذف نویزهای غیرضروری از متن است. این نویزها ممکن است شامل کاراکترهای خاص، اعداد یا علائم نگارشی باشد

مثال: در جمله "سلام!!! من ۲۰ ساله هستم :)", تمامی علائم نگارشی اضافی و کاراکترهای غیرضروری حذف می‌شوند

❖ تبدیل حروف بزرگ به کوچک Lowercasing

برای جلوگیری از وجود چندین نسخه از یک کلمه (مانند "سلام" و "سلام")، تمامی حروف به صورت حروف کوچک تبدیل می‌شوند

مثال: "سَلام" و "سلام" هر دو به "سلام" تبدیل می‌شوند

❖ توکن‌سازی Tokenization

در این مرحله، متن به قطعات کوچک‌تر (کلمات یا جملات) شکسته می‌شود. این قطعات کوچک‌تر به نام توکن‌ها شناخته می‌شوند

مثال: جمله "من به مدرسه می‌روم." به توکن‌های ["من"، "به"، "مدرسه"، "می‌روم"] تبدیل می‌شود

❖ حذف توقف‌واژه‌ها Stopword Removal

کلماتی مانند "و"، "از"، "به" و "در" که به‌عنوان کلمات رایج در زبان شناخته می‌شوند و ارزش تحلیلی بالایی ندارند، حذف می‌شوند

مثال: از جمله "من به مدرسه می‌روم" کلمه "به" حذف می‌شود

مراحل مختلف پیش‌پردازش متن

❖ ریشه‌یابی Stemming یا لِماتیزاسیون Lemmatization

ریشه‌یابی یا لماتیزاسیون فرآیندهایی هستند که کلمات را به شکل پایه‌ای یا ریشه‌شان تبدیل می‌کنند. در ریشه‌یابی، کلمات به فرم ساده‌تری برمی‌گردند، در حالی که در لماتیزاسیون، کلمات به فرم لغوی و معنایی پایه‌شان تبدیل می‌شوند

مثال: "رفتن"، "رفته‌ام"، "می‌روم" همگی به "رفت" تبدیل می‌شوند

اهمیت آماده‌سازی داده‌ها در مدل‌های یادگیری ماشین

مدل‌های یادگیری ماشین کلاسیک معمولاً به داده‌های متنی به‌عنوان ویژگی‌هایی نگاه می‌کنند که به بردارهای عددی تبدیل شده‌اند. این بردارها سپس به مدل‌ها داده می‌شوند. اگر داده‌ها به درستی پیش‌پردازش نشده باشند، نویزها و کلمات زائد می‌توانند دقت مدل‌ها را کاهش دهند. بنابراین، پیش‌پردازش دقیق داده‌های متنی برای بهبود عملکرد مدل‌ها بسیار مهم است

مثال: اگر در یک مسئله دسته‌بندی متون، کلمات غیرضروری و توقف‌واژه‌ها حذف نشده باشند، مدل ممکن است بر اساس کلمات نامرتبط و غیرمفید تصمیم‌گیری کند. این امر دقت پیش‌بینی‌های مدل را کاهش می‌دهد

کاربردهای NLP

❖ ترجمه ماشینی Machine Translation

❖ تحلیل احساسات Sentiment Analysis

❖ خلاصه‌سازی خودکار Text Summarization

❖ پاسخگویی به سوالات Question Answering

❖ تشخیص موجودیت‌های نام‌دار Named Entity Recognition

Regular Expressions یا عبارات منظم

Regular Expressions چیست؟

یک رشته متنی هستند که از الگوهای خاصی تشکیل شده‌اند و برای جستجو و یافتن الگوها در متن استفاده می‌شوند. به عنوان مثال، می‌توانیم با استفاده از یک الگوی ساده، اعداد، حروف بزرگ، یا کلمات خاص را در یک متن پیدا کنیم

کاراکترهای پایه

1. `.` **(نقطه)**: این کاراکتر با هر کاراکتر تکی مطابقت دارد (به جز کاراکتر خط جدید).

- مثال: الگوی `a.b` هر کلمه‌ای که با حرف a شروع شود، یک کاراکتر دلخواه در وسط داشته باشد و با b پایان یابد را پیدا می‌کند؛ مانند "acb" و "a1b".

2. `^` **(علامت شروع)**: برای مشخص کردن شروع خط یا رشته استفاده می‌شود.

- مثال: الگوی `Hello^` رشته‌هایی که با "Hello" شروع می‌شوند را پیدا می‌کند.

3. `$` **(علامت پایان)**: برای مشخص کردن پایان خط یا رشته استفاده می‌شود.

- مثال: الگوی `$world` رشته‌هایی که با "world" به پایان می‌رسند را پیدا می‌کند.

4. `*` **(ستاره)**: به معنای صفر یا بیشتر از وقوع یک الگو است.

- مثال: الگوی `a*` تمامی جاهایی که دارای هیچ یا چندین حرف `a` هستند را پیدا می‌کند، حتی اگر `a` وجود نداشته باشد.

کاراکترهای پایه

5. `+` **(بعلاوه):** به معنای یک یا بیشتر از وقوع یک الگو است.
- مثال: الگوی `a+` هر جا که حداقل یک حرف `a` باشد را پیدا می‌کند.
6. `?` **(علامت سوال):** به معنای صفر یا یک بار وقوع یک الگو است.
- مثال: الگوی `?a` هر جا که حاوی یک یا هیچ حرف `a` باشد را پیدا می‌کند.
7. `{n}` **(براکت‌ها):** برای مشخص کردن تعداد دقیق تکرار یک الگو استفاده می‌شود.
- مثال: الگوی `a{3}` هر جایی که سه بار حرف `a` تکرار شده باشد را پیدا می‌کند.
8. `[ ]` **(براکت‌ها):** برای تعریف یک مجموعه از کاراکترها استفاده می‌شود.
- مثال: الگوی `[a-z]` تمامی حروف کوچک انگلیسی را شامل می‌شود.
9. `( )` **(پرانتزها):** برای گروه‌بندی کردن بخشی از الگو استفاده می‌شود.
- مثال: الگوی `+(ab)` هر جا که حداقل یک بار توالی "ab" تکرار شود را پیدا می‌کند.

قواعد نوشتن یک الگو با مثال

Example number: `+1 (202) 555-1234`

Our regex pattern: `\+?\d{1,3}?[-.\s]?\(?\d{1,4}?\)?[-.\s]?\d{1,4}[-.\s]?\d{1,9}`

1. `\+?`
   - **Purpose**: Match the `+` sign, which often appears as a country code indicator.
   - **Explanation**: `\+` matches the `+` character, and `?` means it is optional (can appear 0 or 1 time).
   - **Application to the example**: In our example, the `+` sign is present at the beginning, so this part matches the `+`.
   - **Match**: `+`

2. `\d{1,3}?`
   - **Purpose**: Capture the country code or prefix, which typically has 1 to 3 digits.
   - **Explanation**: `\d` matches any digit, and `{1,3}` means it can have between 1 and 3 digits. The `?` after `{1,3}` makes it non-greedy, meaning it will match the minimum digits required.
   - **Application to the example**: The number `1` follows the `+`, which matches this part of the pattern.
   - **Match**: `1`

Example number: `+1 (202) 555-1234`

Our regex pattern: `\+?\d{1,3}?[-.\s]?\(?\d{1,4}?\)?[-.\s]?\d{1,4}[-.\s]?\d{1,9}`

3. `[-.\s]?`

- **Purpose**: Identify separators (like dashes, dots, or spaces) between parts of the phone number.
- **Explanation**: `[-.\s]` matches a dash `-`, dot `.`, or whitespace `\s`, and the `?` makes this optional.
- **Application to the example**: After the `1`, there is a space, so this part matches the space separator.
- **Match**: ` ` (space)

Example number: `+1 (202) 555-1234`

Our regex pattern: `\+?\d{1,3}?[-.\s]?\(?\d{1,4}?\)?[-.\s]?\d{1,4}[-.\s]?\d{1,9}`

4. `\(?\d{1,4}?\)?`

- **Purpose**: Match the area code, which may be enclosed in parentheses.
- **Explanation**:
  - `\(?` optionally matches an opening parenthesis `(`.
  - `\d{1,4}` matches between 1 and 4 digits (for area codes or part of the main number).
  - `\)?` optionally matches a closing parenthesis `)`.
- **Application to the example**: The area code `(202)` matches this part, including the parentheses and the digits inside.
- **Match**: `(202)`

Example number: `+1 (202) 555-1234`

Our regex pattern: `\+?\d{1,3}?[-.\s]?\(?\d{1,4}?\)?[-.\s]?\d{1,4}[-.\s]?\d{1,9}`

5. `[-.\s]?`
   - **Purpose**: Another separator between sections of the phone number, similar to step 3.
   - **Explanation**: As in step 3, `[-.\s]` captures a dash, dot, or space, and `?` makes it optional.
   - **Application to the example**: There is a space following `(202)`, so this part matches the space.
   - **Match**: (space)
6. `\d{1,4}`
   - **Purpose**: Match the middle part of the phone number, which may contain 1 to 4 digits.
   - **Explanation**: `\d{1,4}` matches between 1 and 4 digits.
   - **Application to the example**: The next section of the phone number is `555`, which matches this part of the pattern.
   - **Match**: `555`

Example number: `+1 (202) 555-1234`

Our regex pattern: `\+?\d{1,3}?[-.\s]?\(?\d{1,4}?\)?[-.\s]?\d{1,4}[-.\s]?\d{1,9}`

7. `[-.\s]?`

- **Purpose**: Another separator between sections, similar to previous steps.
- **Explanation**: This matches a dash, dot, or space, and is optional.
- **Application to the example**: In the example, a dash `-` follows `555`, matching this part.
- **Match**: `-`

8. `\d{1,9}`

- **Purpose**: Match the final part of the phone number, which can contain between 1 and 9 digits.
- **Explanation**: `\d{1,9}` matches between 1 and 9 digits, covering the last section of the number.
- **Application to the example**: The number `1234` matches this last part of the pattern.
- **Match**: `1234`