冬の3大学合同LT会

正規表現でお手軽冷笑検知

まろん。(@devuloper)

自己紹介

profile.yml

```yaml
name: まろん。
desc: Funny OSS developer
univ: 公立はこだて未来大学
hobbies: [散策, 開発, 温泉巡り, ドメイン]
portfolio: https://montblank.fun/
accounts:
  discord: devuloper
  twitter: rin_montblank
  github: otoneko1102
```

正規表現を知っていますか

正規表現とは(regular expression, regex, regexp)

文字列の検索や置換を効率よく行うための記述ルール

e.g.

\d	一桁の数字
\d+	任意の桁の数字
\d{3}	三桁の数字
\d{3}-\d{4}	郵便番号

メタ文字一覧
https://www.megasoft.co.jp/mifes/seiki/meta.html

正規表現とは(regular expression, regex, regexp)

文字列の検索や置換を効率よく行うための記述ルール

e.g.

.(ドット)	任意の1文字	**()**	グループ化 複数文字対応
^	先頭のみ	**?**	任意 if
$	末尾のみ	*** + ?**	繰り返し 複数検知
|	or	**.***	全て*1

1* 一部例外

メタ文字一覧

https://www.megasoft.co.jp/mifes/seiki/meta.html

正規表現で身近な文字列を検知する

```
[a-z0-9.+-]+@[a-z0-9.-]+\.[a-zA-Z]{2,}
```

正規表現で身近な文字列を検知する

[a-z0-9.+-]+@[a-z0-9.-]+\.[a-z]{2,}

次を(すべて順に)満たしている文字列を検出:

1文字以上の「“a”から“z”, “0”から“9”, “.”, “+”, “-”」のみで構成

「“@”」

1文字以上の「“a”から“z”, “0”から“9”, “.”, “-”」のみで構成

「“.”」(\はエスケープ)

2文字以上の「“a”から“z”」で構成

正規表現で身近な文字列を検知する

[a-z0-9.+-]+@[a-z0-9.-]+\.[a-z]{2,}

次を(すべて順に)満たしている文字列を検出:
1文字以上の「“a”から“z”, “0”から“9”, “.”, “+”, “-”」のみで構成
「“@”」
1文字以上の「“a”から“z”, “0”から“9”, “.”, “-”」のみで構成
「“.”」(\はエスケープ)
2文字以上の「“a”から“z”」で構成

=メールアドレス

まるん rin_montblank

正規表現で身近な文字列を検知する

分かりやすいが冗長

```
const text = "どうもこんにちは私はまろんです";
// 全件をマッピング
const targets = [
    "私は村田です",
    "私はまろんです",
    "僕は村田です",
    "僕はまろんです"
];
console.log(targets.some(target => text.includes(target))) // true or false
```

正規表現で身近な文字列を検知する

コンパクトでわかりやすい

```
1   const text = "どうもこんにちは私はまろんです";
2   // [私僕] -> "私" か "僕" のどちらか1文字
3   // (村田|まろん) -> "村田" か "まろん" のどちらか
4   const regex = /[私僕]は(村田|まろん)です/;
5   console.log(regex.test(text)); // true or false
6
```

また、正規表現のほうが検知が速い場合が多い

冷笑を検知したい

冷笑を検知したい

基本的な冷笑パターンをまとめた正規表現

- どわーw
- うおw😅
- 爆笑爆笑
- …など

```
/**
 * \u
 * 1F605: 😅
 * 1F923: 🤣
 * 203C FE0F: ‼️
 */
const regexStrict: RegExp =
  /(?:\u{1F605}|\u{1F923}|\u{203C}\u{FE0F}?|(?:[きキｷ][ちチﾁ]|[おぉオォｵ][うぅウゥｳ]|[うぅウゥｳ][おぉオォｵ]|[どドﾄﾞ][わゎワヮﾜ])[-ー−～~っッｯ]*(?:[wｗ]+|(?:(?:爆笑)|笑)+|[ （(]笑[） )] )|(?:爆笑){2,}|(?:冷笑){2,}|[ （(]笑[） )])/gu;
/**
 * 1F4A6: 💦
 */
const relaxedOnly: RegExp =
  /(?:[うぅウゥｳ][おぉオォｵ]|[どドﾄﾞ][わゎワヮﾜ])[-ー−～~っッｯ]*|爆笑|冷笑|(?:\u{1F4A6})/gu;
function mergeRegex(r1: RegExp, r2: RegExp): RegExp {
  const flags = Array.from(new Set((r1.flags + r2.flags).split(""))).join("");
  const src = `(?:${r1.source})|(?:${r2.source})`;
  return new RegExp(src, flags);
}
const regexRelaxed: RegExp = mergeRegex(regexStrict, relaxedOnly);
```

※ (?:string)はJavaScriptのString.match()を使う都合の表記

冷笑を検知したい

組み込んで検知するかを検証

まとめ

冷笑を検知できた！
正規表現はすごい🍊
いかがでしたか？

一部のプログラミング言語や検索機能などでは正規表現のパターンや仕様が異なる可能性があるので使う際は注意
(今回は一般的なものを紹介・Node.js v22で動作確認済み)

まとめ

今回作成した正規表現と検知モジュール

https://github.com/otoneko1102/dowa
https://www.npmjs.com/package/dowa

検知モジュールを利用したTwitter用の冷笑チェッカー(ブラウザ拡張機能)

https://github.com/otoneko1102/dowa-twitter-checker

Chromium

Firefox

more: サルにもわかる正規表現入門
https://userweb.mnet.ne.jp/nakama/