

Mesterséges intelligencia alapú spam szűrőfejlesztés neuro-lingvisztikus megközelítéssel, PyTorch keretrendszerrel

Tóth Balázs
Neumann János Informatikai Kar
Óbudai Egyetem
Magyarország, Budapest
mwzx0d@stud.uni-obuda.hu

Dr. Póser Valéria
Neumann János Informatikai Kar
Óbudai Egyetem
Magyarország, Budapest
poser.valeria@nik.uni-obuda.hu

Absztrakt—A tanulmány bemutatja a PyTorch-alapú mesterséges intelligencia spam szűrő fejlesztését, melynek alapját a neuro-lingvisztikus megközelítések képezik, azaz a természetes nyelvi feldolgozás (NLP). Olyan modell került lefejlesztésre, mely segítségével könnyedén ki lehet szűrni a gyanúsnak tűnő üzeneteket.

Kulcsszavak—PyTorch, modell, fejlesztés, NLP, spam

I. BEVEZETÉS

A digitális kommunikáció mára már szinte elengedhetlenné vált az emberek mindennapi életében. Sajnos ezzel párhuzamosan a spam üzenetek terjedése is exponenciálisan növekszik, ezzel kihívást állítva a rendszerekre, amelyek a nemkívánatos tartalmakat szűrik. Kritikus fontosságú, hogy az adott szoftver amit használunk az üzenetek küldésére és fogadására, az megbízhatóan szűrje azokat.

A tanulmány azt mutatja be, hogy hogyan alkalmazható a PyTorch keretrendszer és a természetes nyelvi feldolgozási megközelítések összehangoltan egy intelligens spam szűrő rendszer kialakítására. Képes felismerni és tanulni az általunk megadott bemeneti adatok alapján, hogy azok általános vagy épp gyanús üzeneteket jelentenek-e.

II. MÓDSZERTAN

A. Adatgyűjtés és előkészítés

Első és legfontosabb az adatgyűjtés, hiszen ez az alapja a mesterséges intelligenciának, ezek alapján tud tanulni. A gyűjtött adatok tartalmazzák a spam és nem spam kategóriák reprezentatív mintáit. Az adatforrás felépítése tartalmazza az üzenetet és a besorolást. A besorolás lehetséges értékei 0 és 1, ahol a 0-ás érték jelöli az általános üzenetet és az 1-es érték, hogy gyanús, vagyis spam üzenet. Az adatok előkészítése során a felesleges részeit az üzenetnek el kell távolítani.

B. PyTorch-alapú modell fejlesztése

A modellnek meg kell adni, hogy milyen dimenziójú bemenettel kell számolnia. Belátható, hogy a modellnek három rétege van, melyek teljesen összekapcsolt rétegek,

amik alkotják a neurális hálózatot és a forward függvényen keresztül halad át a bemeneti adatokon. Tegyük fel, hogy az `input_dim` értéke 100, így annak 100 elemű vektorral kell rendelkeznie, ami az első réteget illeti.

```
class TextClassifier(nn.Module):
    def __init__(self, input_dim):
        super(TextClassifier, self).__init__()
        self.fc1 = nn.Linear(input_dim, 64)
        self.fc2 = nn.Linear(64, 32)
        self.fc3 = nn.Linear(32, 2)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Ábra 1. Modell Python kód tartalma

Első réteg (`fc1`), ahol az `input_dim` dimenziójú bemeneteket fogadja és 64 dimenziójú kimenetet generál a rektifikált lineáris egység (ReLU) ($H = \max(0, X)$) aktivációs függvénnyel. A második réteg (`fc2`) 64 dimenziójú bemeneteket fogad és 32 dimenziós kimenetet generál, szintén az előző réteg módszere szerint. A harmadik réteg (`fc3`) 32 dimenziójú bemeneteket fogad és 2 dimenziójú kimenetet generál, így ez a végső réteg, vagyis nem kell további aktivációs függvényt alkalmazni, ezáltal a kimenetet visszaadjuk.

C. Természetes nyelvi feldolgozás integrációja

D. Tanulási folyamat

E. Modell értékelése

F. Finomhangolás és optimalizálás

III. EREDMÉNYEK

IV. MEGVITATÁS

HIVATKOZÁSOK

- [1] Basemah Alshemali and Jugal Kalita, *Improving the Reliability of Deep Neural Networks in NLP: A Review*, Knowledge-Based Systems, 2020.