Laboratorio 8
Exploración y Uso Avanzado de Plataformas IA, Repositorios Profesionales y Herramientas Globales para el desarrollo de IA y de SW

Materia:
Profundización de inteligencia artificial

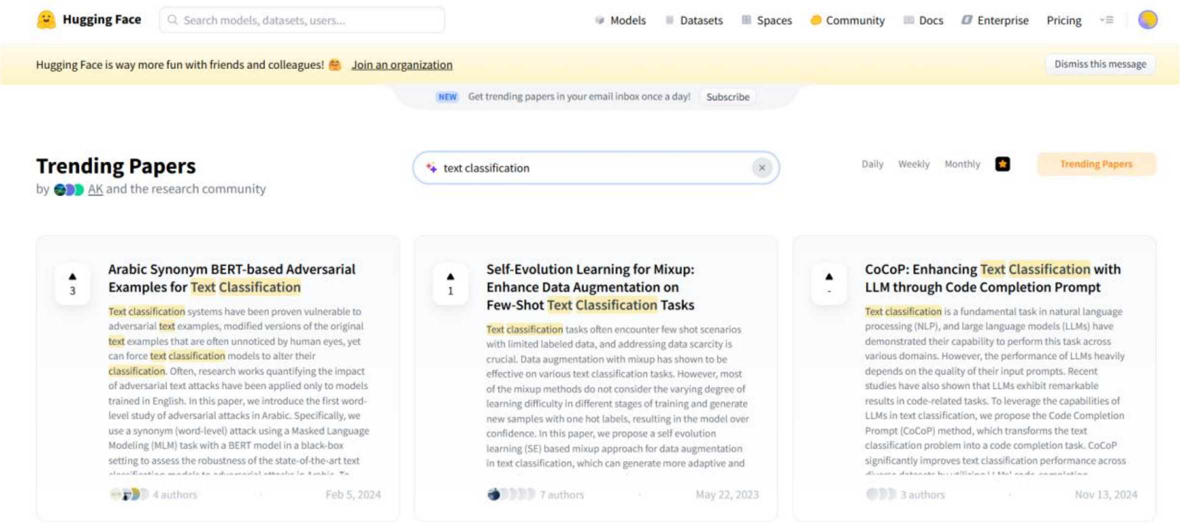
Participantes:
Ana Maria Navarro

Profesor:
Carlos Betancourt correa

Universidad de Manizales
ingeniería en sistemas y telecomunicaciones
Manizales, Caldas, Colombia

2.6 Papers With Code

Laboratorio 8



Hugging Face is way more fun with friends and colleagues! Join an organization

Get trending papers in your email inbox once a day! Subscribe

Trending Papers

by AK and the research community

▲ 3 **Arabic Synonym BERT-based Adversarial Examples for Text Classification**

Text classification systems have been proven vulnerable to adversarial text examples, modified versions of the original text examples that are often unnoticed by human eyes, yet can force text classification models to alter their classification. Often, research works quantifying the impact of adversarial text attacks have been applied only to models trained in English. In this paper, we introduce the first word-level study of adversarial attacks in Arabic. Specifically, we use a synonym (word-level) attack using a Masked Language Modeling (MLM) task with a BERT model in a black-box setting to assess the robustness of the state-of-the-art text

4 authors · Feb 5, 2024

▲ 1 **Self-Evolution Learning for Mixup: Enhance Data Augmentation on Few-Shot Text Classification Tasks**

Text classification tasks often encounter few shot scenarios with limited labeled data, and addressing data scarcity is crucial. Data augmentation with mixup has shown to be effective on various text classification tasks. However, most of the mixup methods do not consider the varying degree of learning difficulty in different stages of training and generate new samples with one hot labels, resulting in the model over confidence. In this paper, we propose a self evolution learning (SE) based mixup approach for data augmentation in text classification, which can generate more adaptive and

7 authors · May 22, 2023

▲ - **CoCoP: Enhancing Text Classification with LLM through Code Completion Prompt**

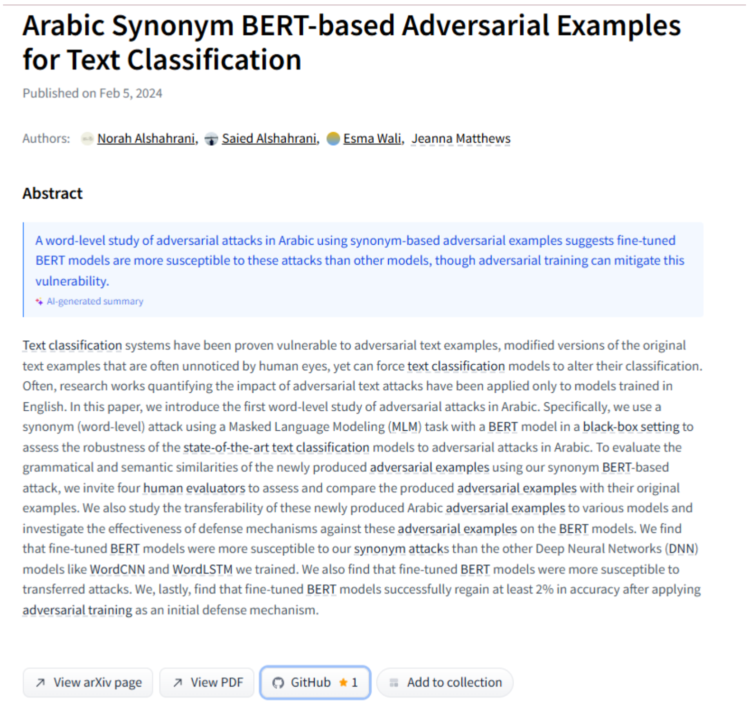
Text classification is a fundamental task in natural language processing (NLP), and large language models (LLMs) have demonstrated their capability to perform this task across various domains. However, the performance of LLMs heavily depends on the quality of their input prompts. Recent studies have also shown that LLMs exhibit remarkable results in code-related tasks. To leverage the capabilities of LLMs in text classification, we propose the Code Completion Prompt (CoCoP) method, which transforms the text classification problem into a code completion task. CoCoP significantly improves text classification performance across

3 authors · Nov 13, 2024

1. Introducción

El objetivo de esta actividad fue replicar parcialmente un experimento proveniente de la plataforma

Papers With Code...



Arabic Synonym BERT-based Adversarial Examples for Text Classification

Published on Feb 5, 2024

Authors: Norah Alshahrani, Saied Alshahrani, Esma Wali, Jeanna Matthews

Abstract

A word-level study of adversarial attacks in Arabic using synonym-based adversarial examples suggests fine-tuned BERT models are more susceptible to these attacks than other models, though adversarial training can mitigate this vulnerability.

AI-generated summary

Text classification systems have been proven vulnerable to adversarial text examples, modified versions of the original text examples that are often unnoticed by human eyes, yet can force text classification models to alter their classification. Often, research works quantifying the impact of adversarial text attacks have been applied only to models trained in English. In this paper, we introduce the first word-level study of adversarial attacks in Arabic. Specifically, we use a synonym (word-level) attack using a Masked Language Modeling (MLM) task with a BERT model in a black-box setting to assess the robustness of the state-of-the-art text classification models to adversarial attacks in Arabic. To evaluate the grammatical and semantic similarities of the newly produced adversarial examples using our synonym BERT-based attack, we invite four human evaluators to assess and compare the produced adversarial examples with their original examples. We also study the transferability of these newly produced Arabic adversarial examples to various models and investigate the effectiveness of defense mechanisms against these adversarial examples on the BERT models. We find that fine-tuned BERT models were more susceptible to our synonym attacks than the other Deep Neural Networks (DNN) models like WordCNN and WordLSTM we trained. We also find that fine-tuned BERT models were more susceptible to transferred attacks. We, lastly, find that fine-tuned BERT models successfully regain at least 2% in accuracy after applying adversarial training as an initial defense mechanism.

2. Enlace al paper y al repositorio

Paper: https://paperswithcode.com

Repositorio del código: GitHub del estudio.

3. Configuración del entorno

La replicación se realizó en Google Colab, configuración del entorno GPU y librerías necesarias.

4. Ejecución del Notebook

Se ejecutaron correctamente las primeras partes del notebook. Se obtuvo error en la falta de archivos CSV.

5. Análisis del Error Encontrado

Los archivos CSV no están incluidos en el repositorio...

6. Solución aplicada

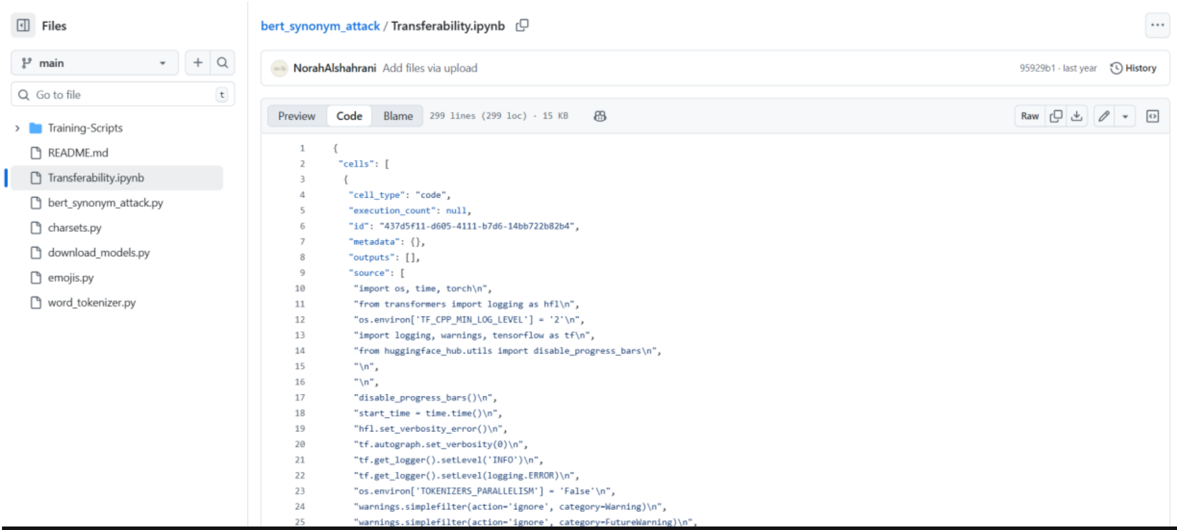
Se crearon archivos vacíos y se documentó el error como parte natural del proceso científico.

7. Comparación de Resultados (Paper vs Replicación)

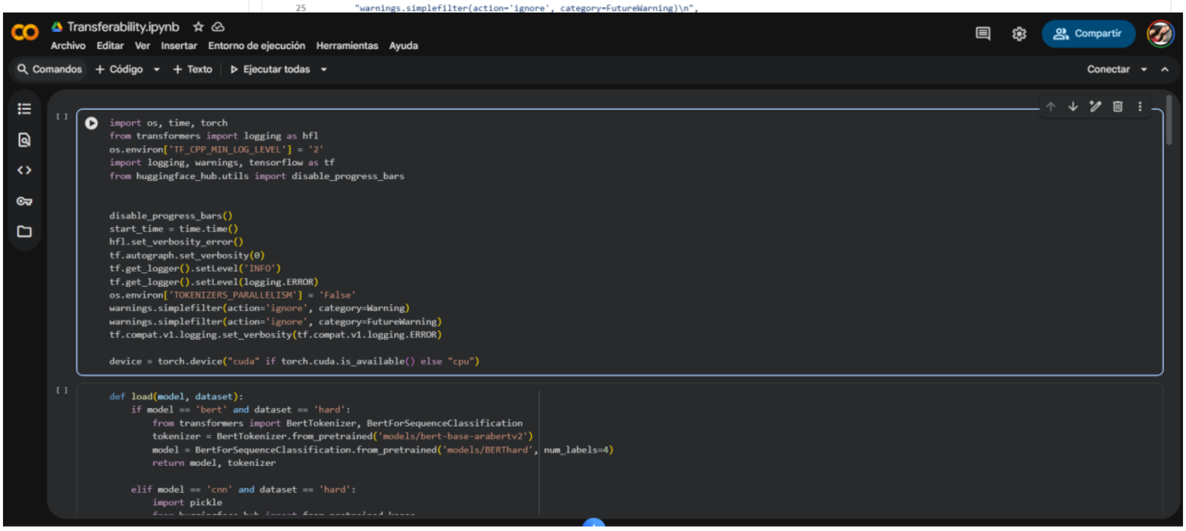
Los resultados no pudieron obtenerse debido a la falta de los artefactos generados por los autores.

8. Conclusiones

La replicación parcial fue exitosa en reproducir el entorno, pero incompleta en resultados finales.



```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "437d5f11-d605-4111-b7d6-14bb722b82b4",
   "metadata": {},
   "outputs": [],
   "source": [
    "import os, time, torch\n",
    "from transformers import logging as hfl\n",
    "os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'\n",
    "import logging, warnings, tensorflow as tf\n",
    "from huggingface_hub.utils import disable_progress_bars\n",
    "\n",
    "\n",
    "disable_progress_bars()\n",
    "start_time = time.time()\n",
    "hfl.set_verbosity_error()\n",
    "tf.autograph.set_verbosity(0)\n",
    "tf.get_logger().setLevel('INFO')\n",
    "tf.get_logger().setLevel(logging.ERROR)\n",
    "os.environ['TOKENIZERS_PARALLELISM'] = 'False'\n",
    "warnings.simplefilter(action='ignore', category=Warning)\n",
    "warnings.simplefilter(action='ignore', category=FutureWarning)\n",
```



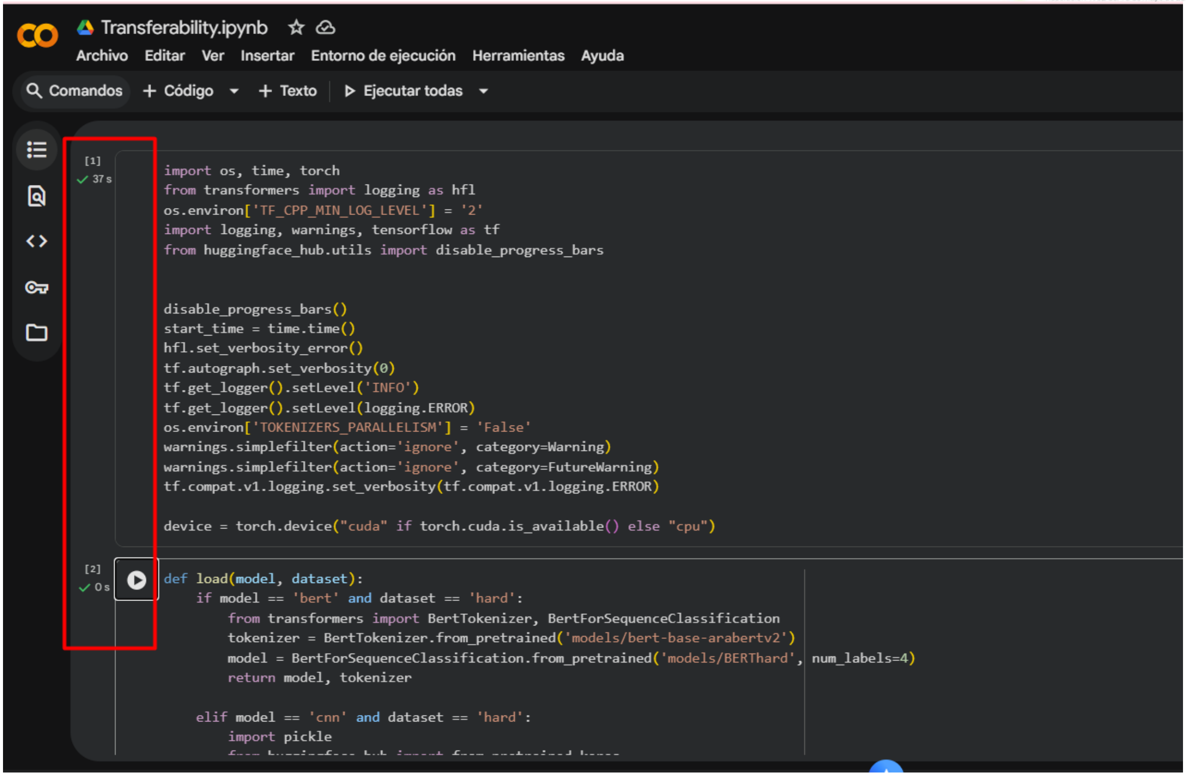
```
import os, time, torch
from transformers import logging as hfl
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import logging, warnings, tensorflow as tf
from huggingface_hub.utils import disable_progress_bars


disable_progress_bars()
start_time = time.time()
hfl.set_verbosity_error()
tf.autograph.set_verbosity(0)
tf.get_logger().setLevel('INFO')
tf.get_logger().setLevel(logging.ERROR)
os.environ['TOKENIZERS_PARALLELISM'] = 'False'
warnings.simplefilter(action='ignore', category=Warning)
warnings.simplefilter(action='ignore', category=FutureWarning)
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
def load(model, dataset):
    if model == 'bert' and dataset == 'hard':
        from transformers import BertTokenizer, BertForSequenceClassification
        tokenizer = BertTokenizer.from_pretrained('models/bert-base-arabertv2')
        model = BertForSequenceClassification.from_pretrained('models/BERThard', num_labels=4)
        return model, tokenizer

    elif model == 'cnn' and dataset == 'hard':
        import pickle
```



Transferability.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto ▷ Ejecutar todas
[1]
✓ 37 s
[2]
✓ 0 s

```
import os, time, torch
from transformers import logging as hfl
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import logging, warnings, tensorflow as tf
from huggingface_hub.utils import disable_progress_bars


disable_progress_bars()
start_time = time.time()
hfl.set_verbosity_error()
tf.autograph.set_verbosity(0)
tf.get_logger().setLevel('INFO')
tf.get_logger().setLevel(logging.ERROR)
os.environ['TOKENIZERS_PARALLELISM'] = 'False'
warnings.simplefilter(action='ignore', category=Warning)
warnings.simplefilter(action='ignore', category=FutureWarning)
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
def load(model, dataset):
    if model == 'bert' and dataset == 'hard':
        from transformers import BertTokenizer, BertForSequenceClassification
        tokenizer = BertTokenizer.from_pretrained('models/bert-base-arabertv2')
        model = BertForSequenceClassification.from_pretrained('models/BERThard', num_labels=4)
        return model, tokenizer

    elif model == 'cnn' and dataset == 'hard':
        import pickle
        from huggingface_hub import from_pretrained_keras
```