A dissertation submitted to the **University of Greenwich**
in partial fulfilment of the requirements for the Degree of

Master of Science
in
Data Science

Detection of Idiomaticity using Transformer Models

Name: Neharika Joshi

Student ID: 001200608

Supervisor: Dr Stef Garasto

Project Deadline: 17th January 2023

Word Count: 13530 words

Abstract

This project presents a study on idiomaticity detection using transformer models. Idiomatic phrases, also known as idioms, are fixed expressions that cannot be understood from the meanings of the individual words. Traditional methods for idiomatic phrase detection rely on rule-based approaches or lexical databases, but these methods have limitations in terms of coverage and accuracy. The use of transformer models and Siamese network architecture offers the potential for improved performance by leveraging large amounts of data and the ability to capture complex relationships between words in two languages. To improve the performance of the models, a stacked ensemble is created using DistilBERT, RoBERTa, and XLNET. The results show that the proposed approach achieved high accuracy in idiomaticity detection, demonstrating the effectiveness of transformer models and the ensemble technique in this task. The goal of this project is to develop and evaluate models that can accurately detect idiomatic phrases in natural language text.

Key Words: Natural Language Processing, Machine Learning, Transformer Models, Idiomaticity Detection, Siamese Network, Idioms, Multiword Expressions

Acknowledgement

My entire project work experience has been a wonderful experience. I started it with having a little knowledge about Natural Language Processing through module ‘Applied Machine Learning’ and some visualizing techniques in ‘Data Visualization’ which was taught in Term 2 (Jan-April 2022). During the term, I found NLP quite interesting with the idea of teaching a machine to recognize the language we speak.

First of all, I would like to thank my partner Deep for his unwavering love, support, and understanding throughout this new struggle in my life without whom, I could not have succeeded.

Also, a special gratitude to my supervisor Dr Stef Garasto who has invested their encouragement, support and counsel by displaying their passion consistently in what they do. Their expertise and patience in answering my questions and providing feedback on my work was greatly appreciated.

Lastly, I would like to thank my parents and family. They have been supporting me throughout my studies and I wouldn’t have gotten this far in my academic career without them.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

- Δ MWE - Multiword Expression
- Δ MWEs - Multiword Expressions
- Δ NLP - Natural Language Processing
- Δ ML - Machine Learning
- Δ IE - Idiomatic Expressions
- Δ PIE - Potentially Idiomatic Expression
- Δ BERT - Bidirectional Encoder Representations from Transformers
- Δ BERTRAM - BERT for Attentive Mimicking
- Δ ROBERTA - Robustly Optimized BERT Pre-training Approach
- Δ XLM-RoBERTa - Cross-Lingual Masked Language Model: Robustly Optimized BERT Pre-training Approach
- Δ SBERT- Sentence Bidirectional Encoder Representations from Transformers
- Δ MT - Machine Translation
- Δ CWE - Contextual Word Embedding
- Δ mBERT - Multilingual Bidirectional Encoder Representations from Transformers
- Δ MLM- Masked Language Modelling
- Δ NSP - Next Sentence Prediction
- Δ GLUE - General Language Understanding Evaluation
- Δ AR - Auto Regressive
- Δ RNN - Recurrent Neural Network
- Δ LTSM - Long short-term memory
- Δ EN - English
- Δ PT - Portuguese
- Δ EN-PT - Both English and Portuguese

1. Introduction

1.1 Overview

Multiword expressions (MWEs) are groups of words that work together as a single unit to convey a distinct meaning that cannot be derived from the words' individual meanings. Based on their structure and function, MWEs can be divided into various categories.

- **Phrasal Verbs:**
  These are verb-particle combinations that convey a different meaning than the individual words, including "turn off," "run out," and "look up."
- **Compound Nouns:**
  These are noun phrases that are made up of two or more words, such as "swimming pool," "heart attack," and "airport terminal."
- **Light verb constructions:**
  These verb-noun combinations, such as "make a mistake," "take a nap," and "deliver a lecture," have a slight or semantically void verb interpretation while the noun bears most of the semantic features.
- **Idioms:**
  A collection of words from a language that represents a metaphorical meaning rather than the literal meaning is called Idioms. Idiomatic phrases are a collection of words that function somewhat independently of one another. It can be both used literally or figuratively which makes it even more complicated especially for beginners while learning a new language. To exemplify, “We have to catch the *big fish* behind this.”, where the idiomatic expression means that there is someone who is the *head of an operation,* and they want to get that particular person for leading the crime, but the literal meaning would be simply catching a *huge fish from the sea*.

Humans can express how they feel through verbal communication (words, texts) or through their facial expressions or body language. Hence, the words of the language play a vital role for expressing. In the current world, text is a powerful source of data with emotional content on the Internet as more and more people are using chatting applications, social media, blog pages, surveys to collect data (Plaza-del-Arco, 2020). This huge

amount of real-time data is just out there collected from various source which is full of emotional information.

Idiomaticity detection has been gaining much attention lately from the NLP community. Creating systems that can automatically analyze natural language in order to comprehend its emotional meaning is an extremely challenging procedure. Mainly, semantic idiomaticity which are MWEs whose meaning cannot be explicitly derived from its constituent. Detecting these idioms in a sentence is still a challenge in the world of Natural language processing (NLP) due to the machine understanding it as their literal meaning rather than the metaphorical meaning. Metaphors are a figure of speech that make an implicit comparison between two unlike things by ascribing a characteristic of one to the other, even though they may have nothing in common on the surface. MWEs can be difficult to identify and comprehend, which makes them a problem for tasks involving natural language processing like part-of-speech tagging, parsing, and machine translation. Idioms actually enhance what people are trying to express, yet they are quite difficult to detect if you don't know about them, which is the case with Natural Language Processing.

Large Transformer-based Language models demonstrate cutting-edge performance on variety of Natural Language Processing (NLP) task; however, they are still a bit behind on capturing multiword expressions. In order to categorise text as idiomatic or non-idiomatic, machine learning is used to train a machine learning model on a labelled dataset containing idioms and non-idioms. This strategy may work well if there is a substantial amount of diverse training data, but it may have limitations if the data is not representative of the kinds of idioms the model will meet in the real world. It is crucial to highlight that due to the variety and originality of idioms as well as the context-dependent character of their meanings, idiomaticity identification can be a complicated task. In order to get superior performance on the task, it may be required to experiment with various models and methodologies.

1.2 Current Scenario

Idioms can give language richness and depth and can be utilized to describe abstract or difficult-to-understand ideas in a clearer, more concise manner. They can be a means of expressing emotions or conveying tone, and they are frequently employed to bring color or humor to language. For instance, the expression "break a leg" is supposed to convey "good luck" and should not be interpreted literally. Idioms have an expressive purpose,

but they can also provide information about the speaker or writer's culture and language. They can offer a window into how people think and communicate as well as reveal the values, assumptions, and beliefs of a culture or group.

Learning a new language can be challenging thing while also be equally rewarding. Although, it can be fulfilling to be able to communicate with people from different area of the world in their own language. So, to completely learn a language, idioms should also be studied. It can be challenging for new language learners because they are unique to the particular language and culture themselves. They often have metaphorical meaning which cannot be described from the meanings of any of words so, they can be difficult to start with if you are learning the language from scratch without having a background knowledge about their cultures.

If it is hard for humans, imagine it being hard for machines. They can learn any language faster than us but when it comes to detecting idioms, they are not quite advanced yet because meaning of it cannot be derived from the idiom itself. Depending on the specific objectives and limitations of the task, there are various methods for developing an ML model for idiomaticity identification. Supervised learning is a popular technique that involves training the model on a sizable, labelled dataset of idioms and non-idioms in the target language. Using the characteristics and patterns it has discovered from the training data, the model may then be used to categorize new text as idiomatic or not. Other methods for idiomaticity detection in NLP include hybrid methods that incorporate supervised and unsupervised techniques, as well as unsupervised learning strategies like clustering. It is also possible to combine ML methods with rule-based strategies, in which idioms in the text are recognized and categorized using a set of rules. Overall, the effectiveness of an idiomaticity detection model will be influenced by the quality and quantity of the training data, the model's complexity, and the particulars of the task. The model must be carefully designed and evaluated for it to function efficiently and consistently.

2. Background

2.1 Literature Review

Idiomatic expressions should have a significant role in NLP as, idiomaticity is a recurring phenomenon which is common in all languages. They are frequently utilized in everyday language to improve fluency and effectively communicate concepts (Baldwin & Kim, 2010). In their paper, Zeng & Bhat, 2021 compared what exactly is the difference between Multiple Word Expressions (MWE), Idiomatic Expressions (IE) and Metaphors. They found that every MWE cannot possibly be an IE, but every IE is a unique kind of MWE which also possess a semantic non-compositionality. Although, Metaphors are a figurative expression that contrasts a characteristic share by two dissimilar entities. Even though some MWEs and IEs use figurative conceptions, not all metaphors are IEs because they are not required to have any of the IE characteristics. It is possible to classify many idiomatic MWEs as lexicalized metaphors. Examples of such circumstances include metaphors that are less conventionalized, arise in innovative perspective (probably in literature) and are not well-established enough to be listed in dictionaries. Moreover, there are few precise tests available for the annotators to distinguish these groups apart, and the difference among them is rarely explicit (Gross, 1982).

However, there is an issue with ambiguity of an expression like 'kick the bucket'. Generally, words like these are often used as idiomatic expressions but they can also be used literally in a conversation. In idiomatic sense, '*Unfortunately, when his uncle finally kicked the bucket, no one agreed to give the eulogy*.' In literal sense, '*Sally was so angry with her mom because made her bathe in the backyard that she kicked the bucket full of water and ran away.'* So, Haagsma, et al., 2020 proposed a new term for representing words like 'kick the bucket' called Potentially Idiomatic Expressions (PIE). PIE means that the word may probably pose as an idiomatic expression, but it potentially can also be used in a literal conversation as well.

In the past decade, there has been a lot of advancement in the deep learning spectrum due to the introduction of Transformer architecture (Vaswani, et al., 2017) and making language translation possible on a whole new level. Furthermore, Multiliginual transformer models can deal with different language as they have access to data from

multiple languages. These models can perform supervised training using methods like mask language modelling, predicting next sentence etc (Devlin, et al., 2019).

A modified version of BERT called BERTRAM (BERT for Attentive Mimicking) was developed, a potent pretrained language model that generated embeddings for new tokens from a limited amount of factors inside an existing embeddings. A form embedding is produced initially by utilizing the previously trained embeddings for every n-grams to further make more embeddings for a phase with a variety of contexts. Schick & Schütze, 2019 achieved this by evalutating NLP models for comprehending rare words spoken by human through swapping out significant terms for unique synonyms. Hence, they proved that BERTRAM outperformed RoBERTA and BERT, which proved the value of their approach. The pre-trained language model like BERT (Devlin, et al., 2019), XLM-ROBERTA (Conneau, et al., 2019), XL-NET (Yang, et al., 2019) are the one that contextual embedding uses the most frequently. These pre-trained models are turning out to be more efficient than using pre-trained word embeddings.

2.1.1 Related Work

The NLP task of Idiomatic Expression (IE) identification is well-known. An MWE-focused shared task was first proposed during the 2008 MWE workshop (Savary, et al., 2017). In the past, neural networks have been used to detect MWE in general but not to identify idioms specifically (Legrand & Collobert, 2016). A survey (Constant, et al., 2017) was taken where they intended to provide information on how MWEs are handled in NLP applications because there was lack of leading concepts to perform this task. The goal here was to study about how MWE processing communicates with the applications like Machine Translation (MT). Nevertheless, this task doesn't take any accountability for the context involved and it might not work on any ambiguous MWEs as PIEs. Furthermore, Rei, et al., 2017 proposed a deep learning architecture to encapsulate the composition of metaphors and assessed the task to detect metaphors. Their finding demonstrated that it is helpful to create a specialized network including a gating function to view the relationship between input and target sources.

(Nandakumar, et al., 2019) tested with contextualized embeddings and non-contextualized embeddings on various levels. Further, they also tested for detecting non-compositionality in MWEs with three levels: word level, character level and document level for. They also calculated the overall composition of the MWE using metrics like direct composition and

paraphrase similarity. For the paraphrase similarity method, they paraphrased the MWE multiple times to represent them in semantic manner and compared the similarity scores if the MWE share a good similarity score with any one of the paraphrased texts. Although, the models were tested with the MWEs by not including their contexts which is probably why their model succeeded for the contextualized embeddings. Hashempour & Villavicencio, 2020 pointed out that including the contexts for each MWEs should be considered before classifying them. Thus, they used the contextual representation of each expression to potentially detect PIEs in the sentence. By using word2Vec (Mikolov, et al., 2013) for non-contextual words embeddings while using both BERT (Devlin, et al., 2019) and context2vec (Melamud, et al., 2016) for the contextualized word embeddings. For the CWE's they tokenized each MWE to a single token, considering the Idiom Principle (Sinclair & Sinclair, 1991) which stated that the language user must have access to a vast number of semi-preconstructed sentences that represent individual options. Their results show that using contextualized embedding with BERT worked well. Due to which recent research has started to concentrate on contextual embedding like for the Task A in (Madabushi, et al., 2022). The goal of this task is to better grasp multi-word expressions using novel classification and sentence similarity tasks. Using a coarse-grained classification into an "Idiomatic" or "Non-idiomatic" class, this task assess how well models can detect idiomaticity in text.

3. Implemented Algorithms

(Madabushi, et al., 2022) introduced a Baseline system along with their paper which used mBERT as model which was finetuned to their requirements. In my report, I will be comparing the results of the Baseline system with the rest of the models that I used including Siamese Network implemented models.

Transformers

A neural network detects the relationships between the sequential text inputs and learns the context in order to give the output is known as the Transformers (Vaswani, et al., 2017). Like most of the neural networks, it consists of an encoder block and a decoder block. The Nx in the Figure 1 represents that the x layers can be stacked up into both the encoders and decoder. They operate by utilizing multiple mathematical approaches like Attention to find relationship in each data. This approach has three new approaches: Positional Encoding, Attention & Self-Attention. Positional encoders are used for tagging the position of each word in the sentence which are followed by the multi-head attention layer and Feed forward layer. For every word passed in the attention layer has their own attention vectors which captures the essence of the contextual relationship between the

texts. Furthermore, the attention vectors are forwarded to the feed forward to transform it into a suitable format to pass it to the next encoder or decoder block.

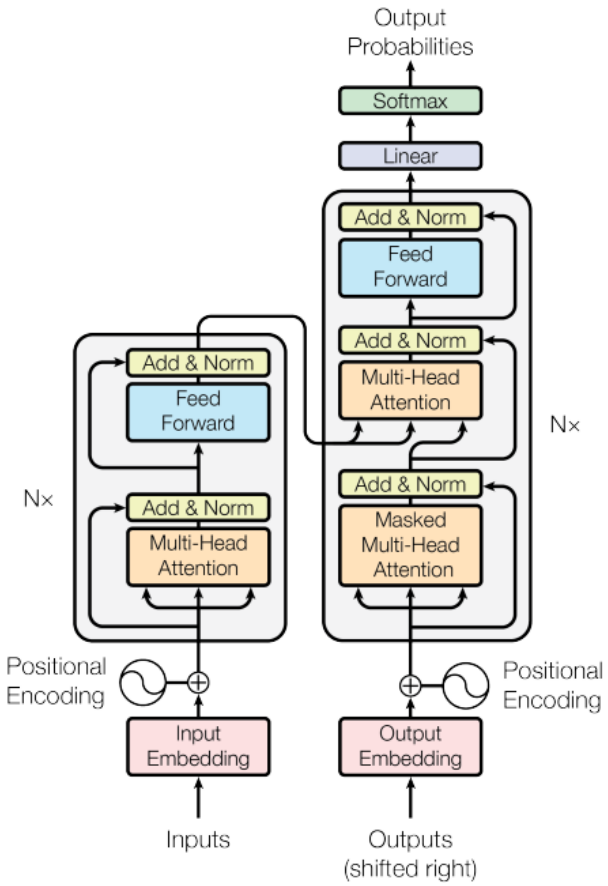

Figure 1: Transformer Model Architecture [Source: (Vaswani, et al., 2017)]

Contextual Embedding

Unlike traditional word embeddings such as word2vec and GloVe, which builds a fixed representations for every word from a large vocabular. Contextual embedding produces a derivative of word embedding depending on which contextual sense is the word appearing as in a sentence. Transformer models generate contextual embeddings to generate a proper representation of a text in a sentence with respect to the other words in the sentence as well. To exemplify, let's consider a phrase 'I have a mouse and a hamster, but I like my hamster better'. Here, the vector representation of the twice occurring word 'hamster' using a traditional word embedding would be same regardless of what the sentence is trying to convey. In contrast, when a pre-trained transformer model, preferably BERT generated a contextual embedding. Hence, the word 'hamster' would have different vector representation while considering the context in which it may appear. In the first context,

hamster is used to say that the person has two pets and one of them is a hamster. In the second context, 'hamster' represents that the out of two of the pets, the person prefer hamster than the mouse.

3.1 mBERT (Baseline)

mBERT stands for Multilingual Bidirectional Encoder Representations from Transformers. It is a deep learning model that has been trained in different languages. mBERT is a derivative of BERT that can carry out NLP tasks in various languages in which it has been trained on. It was pre-trained using a dataset created from variety of Wikipedia texts in over 100+ languages which includes English, Italian, Spanish, Chinese,

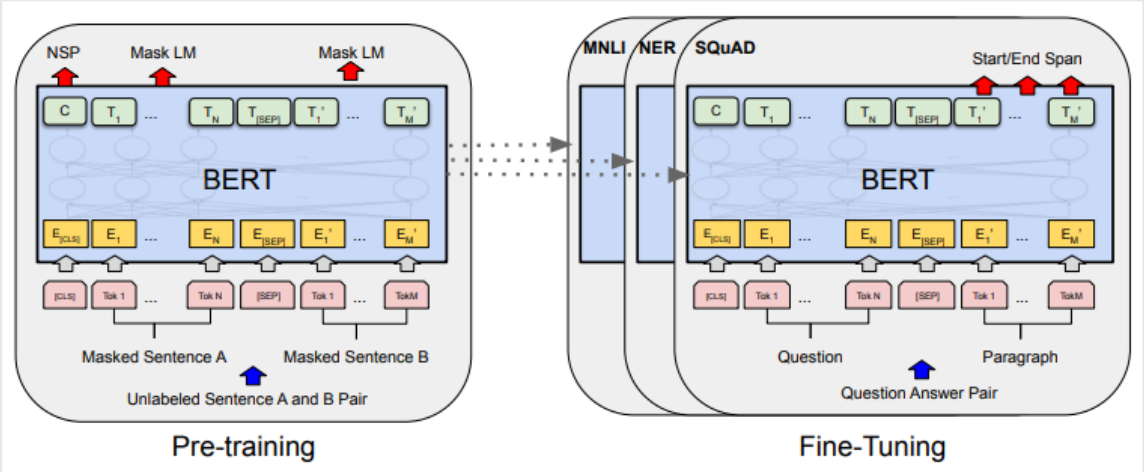

Figure 2: Pre-training and Fine-tuning procedure of BERT [Source: (Devlin, et al., 2019)]

French etc. BERT is the first bidirectional, unsupervised language representation while only being trained using the texts from Wikipedia (Devlin & Chang, 2018). This makes mBERT suitable to perform tasks that involve multiple languages or also tasks that require cross-lingual understanding since it enables itself to learn the patterns and similar characteristics seen across languages. One of mBERT's key feature is that it can perform bidirectional processing which means it uses analyzes the context of words from both left and right directions in a text. This innovation allows BERT to catch a better picture of what the words/sentences really mean, which is one of the crucial parts in most of the Natural Language Processing (NLP) tasks. Applications of mBERT are cross-lingual document classification, machine translation where BERT has proven to be quite effective. It can be used as a pretrained model or can be fine-tuned for any specific tasks.

3.1.1 Masked Language Modeling (MLM)

It is a process which consists of masking out a portion of input tokens and then predicting what the masked tokens further will be on the given context with the rest of the input. Its main objective is to predict the original word using the context that the other words in the

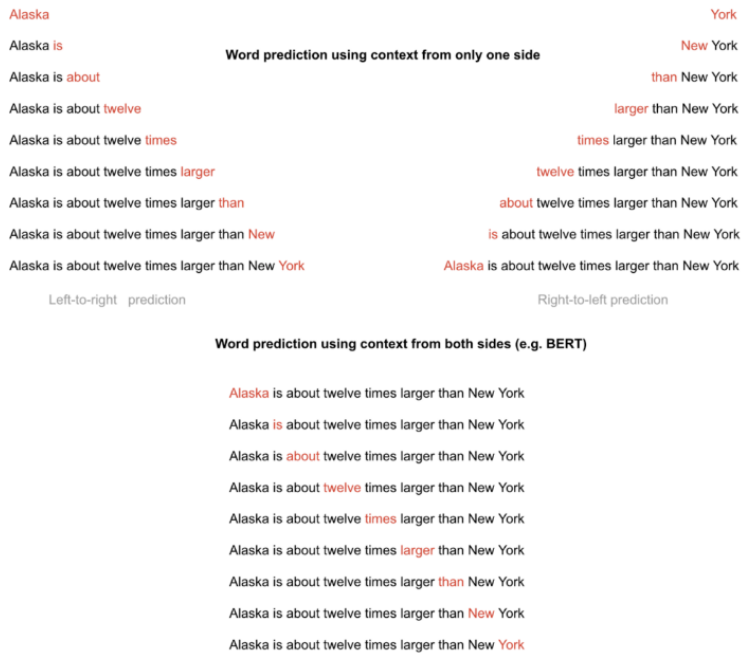

Figure 3: How Masked Language Model works? [Source: (Verma, 2022)]

input give. Large transformer-based language models, like BERT, are frequently trained using MLM to comprehend the meaning and context of individual words within a sentence. BERT uses input sequences by employing mechanism of self-attention. The model is normally given a sentence in masked language modelling, but some of the words have been substituted by a unique placeholder token (e.g., [MASK] or [CLS]). About 15% of word sequences are replaced with [MASK] token before being fed to the transformer model. By using information provided by the other words within the sentence, the model is then taught to forecast the original word that was masked. For example: In sentence ‘The [MASK] is huge animal’, using the provided information the model may well be trained to guess the word ‘elephant’ according to the context offered by the word ‘huge’, implying that the [MASK] token might relate to a heavy animal.

Furthermore, while discussing the model’s operation, it should be considered that the model must be taught the statistical characteristics of word sequences as it may be required to predict one or more words but not the whole phrase.

A model can be initialized with a good concept of linguistic structure and context by pre-training it on masked language modelling. Later, it can be even fine-tuned for a specific task as well. One of the main benefits of this technique is that it enables that model to study contextual links between words in phrase. For transformer-based language models, which heavily depend on attention mechanisms to comprehend the context and meaning of words in a sentence, implementation of MLM especially helpful.

3.1.2 Next Sentence Prediction (NSP)

Next Sentence Prediction (NSP) is also a step for pretraining the model. It is the process that entails determining whether two sentences are linked. To help language models comprehend the link between sentences in a document, it is frequently employed as a training task. This aids the model's ability to recognize how sentences are interrelated with one another and how to use the context used in previous sentence to understand the next. Any given NLP technique aims to comprehend spoken human language in its natural setting. For BERT, this often entails picking a word out of a blank. Models must typically be trained using a sizable collection of specific, labelled training data to accomplish this. This process in designed as a binary classification task. The outcome of BERT representations encapsulates a representation of forthcoming content and contain discourse expectancy representations that are necessary for categorizing coherence relations (Devlin, et al., 2019). The model is trained to determine whether the second sentence is the next logical sentence in the text after being given two sentences. For instance, the model would be taught to predict that the words "The dog had its tongue out" and "The butterfly was colorful" are unrelated. On the other hand, the model would be trained to predict that the words "The dog had its tongue out" and "It had been a hot day" are connected given only those two lines.

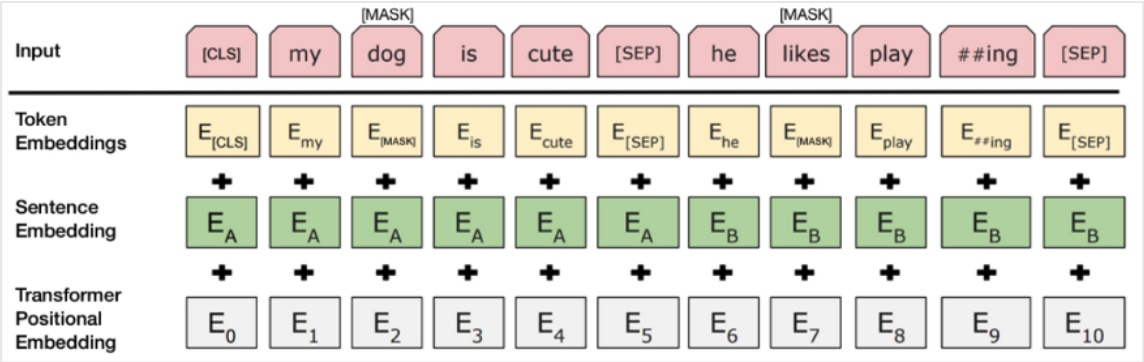

Figure 4: Representation of input in BERT [Source: (Devlin, et al., 2019)]

Before starting with the model, the input is processed as Figure 4 which helps the model to differentiate between any two sentences. The first sentence has a [CLS] token at the start, followed by [SEP] token which separates each sentence. Every token has a sentence embedding that designates it as Sentence A or Sentence B. These embeddings share a similar notion to the token embeddings. Each token receives a positional embedding to denote its place in the sequence.

3.1.3 A close study of mBERT

Back in 2014, the task of predicting the true meaning of a MWEs, especially in a non-English language has received low attention (Salehi, et al., 2014). However, speaking as of 2022 there has been a lot of development in other languages. In 2020, some researchers in Sorbonne University (Paris, France) created a BERT model called CamemBERT(Martin, et al., 2020) specifically for French which showed a remarkable improvement compared to mBERT and other multilingual languages. This revolutionized creating a language specified BERT. To exemplify: BERT Japanese (from Japan), RuBERT (for Russian), ALBERTo (for Italian) etc.

mBERT in general was trained using 104 languages from around the world which includes languages like English, French, Nepali, Japanese, Hindi, Newari etc. Libovický, et al., 2019 performed research in Charles University (Prague, Czech Republic) regarding the components that make mBERT language-neutral and language-specific and discovered that mBERT grouped languages into groups. Furthermore, it interpreted identical texts in many languages in several ways. Thus, that's why mBERT exceptionally well in limited languages (generally English, French) as opposed to creating a language-neutral concept.

3.2 XLNet

XLNet was introduced in 2019 by (Yang, et al., 2019). Back then, it was one of the most advanced models to be introduced for Natural Language Processing. The model provides innovative performance for any common NLP tasks that makes up the GLUE benchmark when it is trained with a huge NLP corpus. Recent developments have been inspired a lot from the unique technique that XLNet uses. XLNet is also inspired from the transformer model BERT (Devlin, et al., 2019) although it implements quite a few advanced techniques than BERT. To simply put it, XLNet is a Generalized Auto-Regressive pretraining model unlike BERT that uses Auto-Encoding language model. When training, rather than just scanning left to right of the target token, it determines the likelihood of a word token based on token combinations in a phrase. AR model performs pretraining by increasing the chances using the forward autoregressive factorization:

$$\max_{\theta} \log p_\theta(x) = \sum_{t=1}^{T} \log p_\theta(x_t | x_{<t}) = \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(x_{1:t-1})^T e(x_t)\right)}{\sum_{x'} \exp\left(h_\theta(x_{1:t-1})^T e(x')\right)}$$

where $h_{\theta(}x_{1:t-1)}$ is a context representation produced by neural network models like Recurrent Neural Networks or a Transformer model and e(x) denoted the embedding of x (Yang, et al., 2019). AR models is ineffective at simulating complex bidirectional contexts as it is only trained to encode in a single direction either backward or forward. Generally, basic language understanding usually prefer bidirectional context information which is one of the disadvantages of AR models.

Although, models like BERT use Auto-Encoding Models that uses [MASK] tokens. Hence, the ≈ in the equation below represents that Auto-Encoding Model's bidirectional property. It randomly masks(x) into a corrupted version $\hat{x}$ hence, the training objective is to reconstruct from $\bar{x}$ to $\hat{x}$:

$$\max_{\theta} \log p_\theta(\bar{x} | \hat{x}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t | \hat{x}) = \sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{x})_t{}^T e(x_t)\right)}{\sum_{x'} \exp\left(H_\theta(\hat{x})_t{}^T e(x')\right)}$$

where $m_t = 1$ which says that x_t is masked and H_θ is a transformer model. It maps a length(T) text sequence(x) into a series of hidden vectors $H_\theta(x) = [H_\theta(x)_1, H_\theta(x)_2 \ldots H_\theta(x)_T]$. The encoding symbols like [MASK] which is used by BERT is not present during pretraining from real data at the time of finetuning, this results in difference between

finetuning and pretraining. However, the predicted tokens are altered (due to masking) so calculating the joint probability to use product rule is not possible for BERT as per AR language modeling.

3.2.1 Architecture

XLNet consists of two phases: pre-training and fine-tuning. It primarily focuses on pretraining phase where it uses a new objective called Permutation Language Modeling. As from the name itself it's clear that the model uses Permutation for processing. Suppose there exists T! unique orders that can be uses to carry out a legitimate autoregressive factorization of a series(x) of length(T). The objective is:

$$\max_{\theta} \ \mathbb{E}_{z\sim Z_T}\left[\sum_{t=1}^{T} \log p_\theta\left(x_{z_t}\middle|\, x_{z_{<t}}\right)\right]$$

Where Z_T is the set of all the possible permutations of the sequence.

Let's consider there to be sequence (x1, x2, x3, x4). Hence, for these 4 tokens there can be possibly 24 permutations. So, for predicting the 3rd token, the pattern that requires 4 possibilities would be x3 at 1st,2nd,3rd and 4th position.

$$\begin{bmatrix} x3 & . & . & . \\ . & x3 & . & . \\ . & . & x3 & . \\ . & . & . & x3 \end{bmatrix}$$

Here, as the x3 token is placed diagonally for all possibilities. Every word and length in the sequence of the words before x3 are present. The model will instinctively learn to acquire data from all viewpoints on both sides. This mechanism was adapted from Transformer-XL (Dai, et al., 2019).

As a result, XLNet doesn't have a support system. The settings given make it difficult and occasionally unclear to determine if a word is in a sentence or not. This enables it to extract additional knowledge from the training corpus. XLNet does not see every relationship because it samples from all combinations. Additionally, it avoids using tiny contexts, which are known to impair training. Following the application of these useful heuristics, it resembles BERT more.

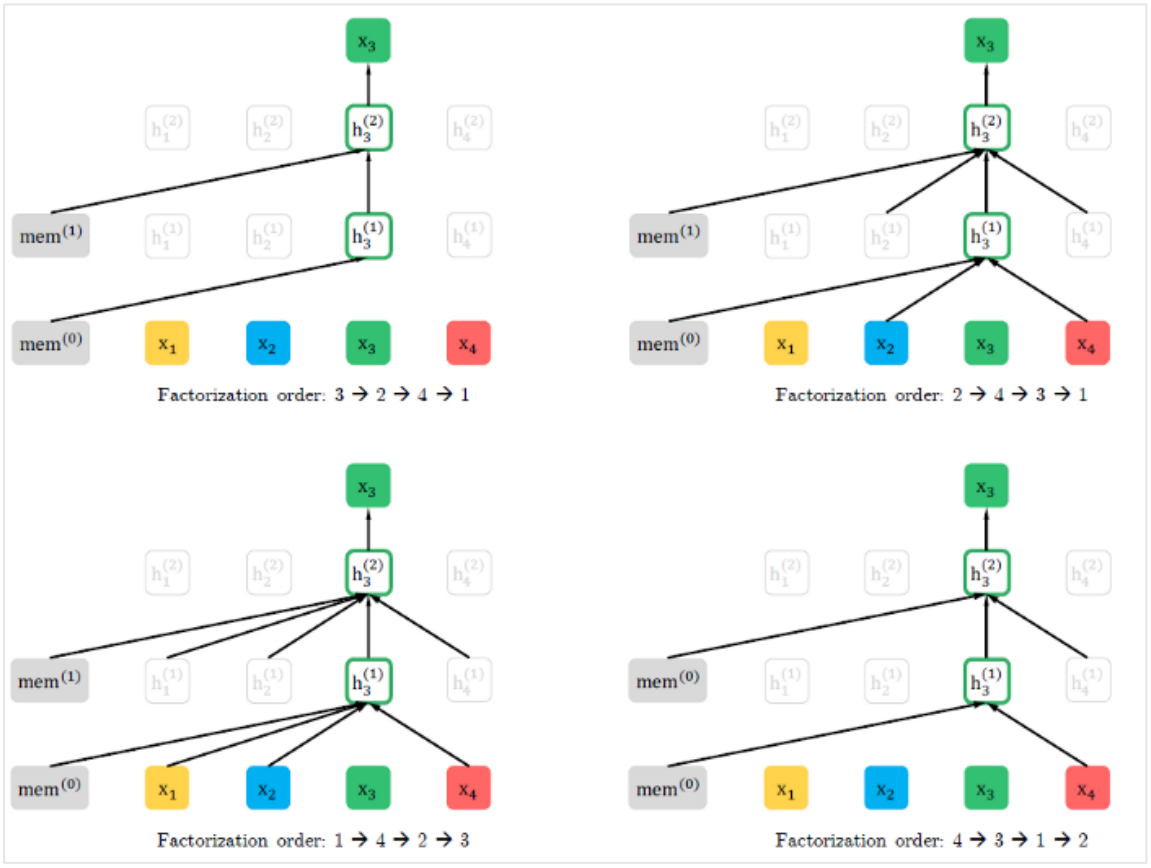

Figure 5: Example of Permutation objective to predict x3

3.3 DISTILBERT (Multilingual)

DistilBERT (Sanh, et al., 2019) is a tinier, faster, cheaper and lightweight version of BERT (Bidirectional Encoder Representations from Transformers), which is a transformer-based model developed by Google for Natural Language Processing Tasks. It was developed by Hugging Face and based it on the architecture of BERT. Distilling Knowledge (Hinton, et al., 2015) is to squeeze a larger model or a collection of models to mimic it and create a compact version of it. This is how the DistilBERT was created. Like BERT, DistilBERT is also a pre-trained on a huge corpus of textual data with the use of Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). As a result, the model can acquire representations of general-purpose languages that can be tailored for tasks. One of the major differences between BERT and DistilBERT is that it is smaller in size, and it has almost 66% lesser parameters than BERT, which is a substantial difference. Furthermore, it can be said that DistilBERT is faster for training and extremely easy to deploy the product. Regardless of its size, this model has been a remarkable to

perform NLP tasks making it a strong option for applications that needs efficient and faster results. DistilBERT is better suited for real-time applications like online chatbots or language translation since it has a faster inference time than BERT.

Student Architecture:

After investigating, it was found that the variations on the last dimension of the tensor have less of impact on efficiency of computation than the variations on other factors like number of layers, because majority of operations used in the architecture (linear layer and layer normalization) are highly optimized. So, including fewer layers in the model was the major priority (Sanh, et al., 2019).

Student Initialization:

Finding the best initialization for the converging sub-network is a vital stage in the training process, in addition to the architectural decisions and optimization that have already been covered. In this manner, the student network is initialized from the teacher network by taking one layer out of two, taking use of the shared dimensionality between the teacher and student networks (Sanh, et al., 2019).

3.4 XLM-RoBERTa

3.4.1 RoBERTa

RoBERTa (Robustly Optimized BERT Pre-training Approach) is a BERT extension that overcomes a few of the original BERT model's drawbacks. RoBERTa performed better on a variety of NLP tasks than BERT since it was trained on a much larger dataset. RoBERTa was implemented in PyTorch, and it includes disregarding BERT's next sentence pretraining objective. The major parameters in BERT are modified, training is done with substantially bigger mini-batches and learning rates. As a result, RoBERTa outperforms BERT in terms of the masked language modelling and performs better on subsequent tasks. RoBERTa is also trained extensively with more data than BERT over a longer period. The findings of the research done by Facebook AI demonstrates that technique for training BERT may be tuned to considerably enhance its performance on a range of NLP tasks while also demonstrating that this general strategy is still competitive with other strategies. Additionally, RoBERTa modified the training process by utilizing a bigger batch size, a longer training period, and dynamic masking, which enables the model to learn to use various amounts of context based on the job. More generally, this study

shows that self-guided training methods can perform as well as or better than more conventional, supervised methods (Meta AI, 2019). Furthermore, RoBERTa is trained on a huge dataset that contains more than 160GB of uncompressed text, as opposed to only 16GB dataset that was originally used to train BERT. The size of the model and the number of parameters vary between the RoBERTa-base and RoBERTa-large versions, which are both accessible.

XLM-RoBERTa (Cross-Lingual Masked Language Model - Robustly Optimized BERT Pre-training Approach) is a cutting-edge cross-lingual language model created by Facebook AI. It is a variation of the RoBERTa (Robustly Optimized BERT Pre-training Approach) model which is already a variant of BERT (Bidirectional Encoder Representations from Transformers) model. XLM-RoBERTa (Conneau, et al., 2019) is trained on a sizable dataset of text in numerous languages and its effective on several natural language processing tasks. It has demonstrated reliable performance on a number of benchmarks, including language production, summarizing, and machine translation. The capacity of XLM-RoBERTa to perform effectively on a range of languages, even when trained on a single language is a remarkable milestone. This open doors for applications consisting numerous languages or requiring language generalization in natural language processing.

3.5 Siamese Network

In the field of supervised machine learning, there is a term called '**Similarity Learning**'. Its main objective is to create a similarity function that calculates the relationship between two elements returning a similarity value. If they are similar, the score would be higher and if not, the score will be lower comparatively. To exemplify: Banks always use signatures comparison to confirm whether the cheque is genuine or not. So, with the help of similarity learning, a model can be trained to recognize if the client's signature is real or forged. If it's real, then the cheque can be passed through but flagged if the signature is forged. Regarding this topic, research was conducted by (Bromley, et al., 1993). They found that the model enhanced resistance to forgeries for regular signers, which was independent from the direction they signed with less sensitivity to the variation in inclination or size of the signature. It performed well with signatures from people with European, American and Chinese background.

Similarly, in NLP, similarity learning can be applied. The similarity between two or more texts can be analyzed by calculating their similarity score. One of the most basic places to apply this concept into is to determine the duplicate questions/topic in popular discussion websites Reddit, Quora, Answers or StackOverFlow. It might seem easier to compare the questions if the words in it are like one another like ‘Where are you from?’ and ‘Where did you come from?’ Although, there might be questions which are similar, but the words can be completely different. To exemplify, ‘How long have you been to London?’ (Sentence A) and ‘When did you first arrive to the Big Smoke?’ (Sentence B) are questions with the same idea. London is called ‘The Big Smoke’ by the rural British people back in 1950’s due to burning of coals and smoke from the industrial factories which killed 1000’s (Barnaby, 2022). Moreover, Sentence A and B are identical questions but differ in choice of words. Thus, training a Siamese Neural Network to identify these aspects of similarity is ideal.

A neural network architecture known as a "**Siamese network**" consists of two or more subnetworks that are identical to one another and have the same weights and biases. Siamese networks are used for tasks like speaker verification, face recognition, and signature verification that require comparing or figuring out how similar two inputs are. In the context of metric learning, when the objective is to train a distance function that can assess the similarity between two inputs, Siamese networks are frequently used. In Siamese networks, the distance between similar inputs is minimized while the distance between different inputs is maximized. The network's output is a distance metric that may be used to assess how similar the inputs are. The fact that the subnetworks in Siamese networks have the same weights and biases is a key characteristic. Since all the inputs are processed using the same set of parameters, the network may develop a universal representation of the data.

Siamese Network guarantees the accuracy of its forecasts. Each network calculates the same function and, the linking weights confirms that two similar image or texts probably won’t be mapped as drastically different in feature space by their individual networks. The network is symmetrical, thus if we give the twin networks two different images, the top conjoining layer will calculate the same metric as if we gave the opposite twins the same two photos. The networks are identical so, even if it is fed with two completely distinct image/text, the upper layer will calculate the same metric as if the two opposite networks have been fed the same image/text (Koch, et al., 2015). Hence, the networks

enable a comparison between the highest-level feature representations on both sides and their parameters are the same. In Figure 6, two hidden layer (Siamese network) with

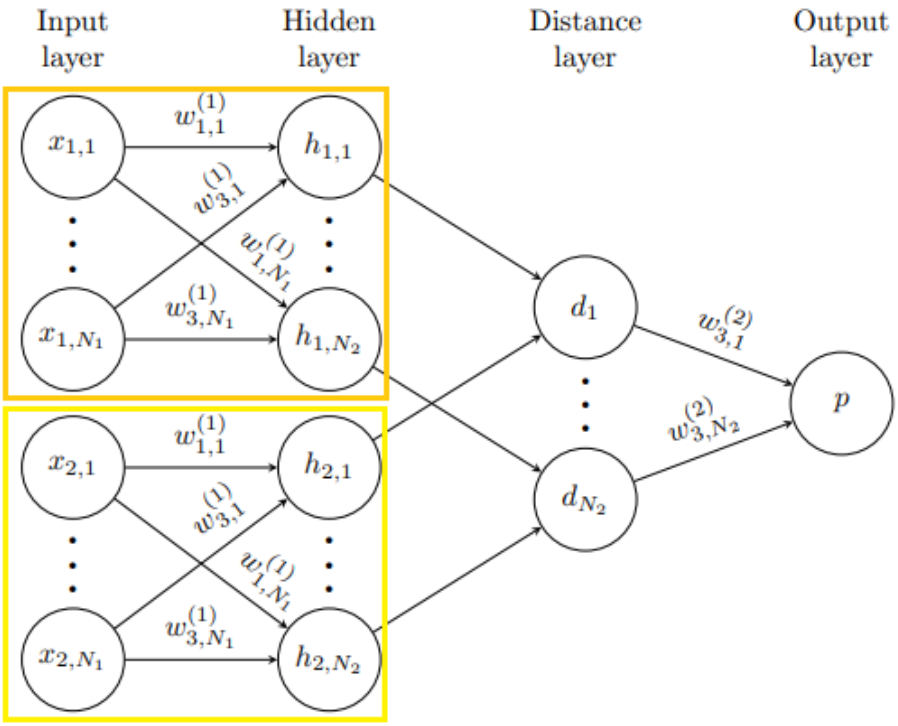

Figure 6: Siamese Network in Hidden Layers of Neural Network [Source: (Koch, et al., 2015)]

logistic prediction(p) is used for binary classification. Twin networks (highlighted layers) are created by duplicating the network's structure in the top and bottom regions and share weight matrices at every layer.

3.5.1 SBERT

The main aim of semantic search is to find sentences that has a similar meaning to each other. Transformers like BERT, RoBERTa performs task like semantic search by computing the word-level or token-level embeddings which cannot be the right way to investigate it because a sentence needs sentence-level embeddings when creating sentence vectors. Furthermore, when it comes to comparing large sentences (probably paragraphs of sentences), using transformers like BERT requires feeding both the sentences into the system which takes lots of time to process (about 65 hours). Hence, it isn't scalable.

So, to overcome the drawbacks of using BERT for comparing the similarities in texts Reimers & Gurevych, 2019 came up with Sentence-BERT (SBERT). It is a modified version of pre-trained BERT network which uses siamese and triplet network to compare

the sentences by calculating cosine similarity. It also uses lost function like softmax loss, cross entropy loss to create identical embeddings for similar texts and non-identical for not so similar texts.

4. Analysis of the System

4.1 Legal, Social, Ethical and Professional Issues

Different moral principles, perceptions, and restrictions on what exactly is the right or wrong thing to do impacts our day-to-day life. In the past, most of the natural language processing (NLP) applications focused on enhancing text that was not strongly associated with a specific author and was often published publicly and with a considerable time lag. These factors created a separation between the text and the author, which prevented the research from having a direct impact on the author's situation. So, when discussing about NLP in the same context, there are some ethical concerns that arise which requires a proper investigation. Despite being an effective tool for both organizations and individuals, NLP has drawbacks and its own set of restrictions to overcome.

The process of idiomaticity detection requires locating and examining idioms, which are words with figurative and nonliteral meanings. Because idioms can significantly alter a text's meaning, idiomaticity identification is crucial in NLP. For instance, incorrect machine translation of an idiom can produce a translation that is misleading or perplexing. Similar to this, when classifying texts, the use of idioms might influence which group they fall into. Identifying idioms can lead to a number of moral and social problems. The data that needs to be collected for the analysis for this task may purely consist of personal data which raises questions regarding the privacy of the consumers and chances of data exploitation. Idiomaticity detection algorithms that produce biased conclusions based on the language employed in texts might reinforce and exaggerate preexisting biases. When a particular segment of the population is either overrepresented or severely underrepresented in the data, it happens. It causes the models to make poor inferences and erroneous generalizations. The employment of idiomaticity identification technologies to censor or prohibit types of content raises questions regarding the freedom of speech as well.

The language of the data used also impacts on the ethical and social elements of detecting idioms. Text processing majorly focuses on the mostly spoken languages in developed European countries rather that small ethnic languages from Asia or Africa. This only creates huge amount of data and analysis in a selected set of languages which results to imbalance in the collected data (Hovy & Spruit, 2016). This causes the underexposure of

rest of the languages. English is one of the most spoken languages in the world, even used as a common language to communicate. Hence, more and more systems are being developed in English as the market in English is in high demand and profitable. But recently multilingual algorithms are also being introduced which may open possibilities for analysis in other languages.

Idiomaticity detection systems can spread misinformation, particularly when they are used to automate the creation or dissemination of content. But there may be cases where the same idioms can be used by different people for a different meaning. So, is it possible that we are prioritizing Artificial Intelligence over people and their use of idioms? Teaching machine to understand multi word expressions won't do anybody any harm unless the MWE's are translated using language translators where, the idioms may be represented differently. Interestingly, I have noticed this while using Facebook translate function that the English idiom "Hang in there", when translated to Hindi "*वहाँ पर लटकले*" pronounced "waha par latakle" which sounds a bit offensive to those who only understand Hindi and might think that the phrase literally means someone telling "to hang themselves". The MWE conveys exactly opposite of what the MWE is trying to express. Hence, idiomaticity detection algorithms have the potential to distribute false information, especially if they are used to automate the production or distribution of content.

This is a rapidly evolving field, and professionals working in this area need to stay up to date with the latest developments and best practices for the best results. So, when building an idiom detection system, the results obtained could be inaccurate or unreliable that might have major repercussions if they are used for decision-making procedures. Idiom detection projects often involve working with a range of stakeholders, including domain experts, data scientists, and software developers, and it is important for professionals to be able to collaborate effectively with these groups.

5. Exploratory Data Analysis

5.1 Data

The data is collected by (Madabushi, et al., 2021). It consists of 8 columns:

- DataID: It represents a unique number of the row.
- Language: It determines whether the given row has English or Portuguese language.
- MWE: It determines which of the MWE is present in the sentence.
- Setting: This represents which setting(one-shot/zero-shot) is the data for.
- Previous: This given the sentence that comes before the sentence that consists of the MWE.
- Target: This column contains the sentence that has the MWE present in it.
- Next: This column consists of the sentence that comes after the sentence that consists of the MWE
- Label: It determines whether the sentence is Idiomatic (0) or not (1).

The main dataset is further divided into 2 parts, training, and testing sets. Furthermore, two setting is available:

i. **Zero-shot:**
   In this scenario, the training set will not contain any of the MWE's that will be tested with the development set. Hence, the prediction will made for the MWE's that the model will not have seen beforehand. Also, for this setting, the previous and next context for the target sentence will be provided in both training and testing set.

ii. **One-shot:**
   In this scenario, every MWE present in the training set will be present in the development set so that there will be some familiarity. Also, idiomatic and non-idiomatic of each MWE will be present in this setting as well.

5.1.1 Why One-shot and Zero-shot setting?

It is quite common in ML to use one-shot and zero-shot learning techniques to train models with finite amount of labeled data.

Zero-shot learning is the process where just by using it's knowledge from the training data, the model can classify new samples from the classes that it hasn't seen before. It's used in situations where there is less amount of data for a certain class which in our case is the literal use and metaphorical use of the same MWEs.

One-shot learning is a technique that allows the model to categorise new instances from classes for which it has only seen one or a very small number of examples. This is helpful when gathering labelled data for certain classes is expensive or complex, or when the model has to quickly identify and categorise new classes.

5.2 Data Collection

As per (Madabushi, et al., 2021), twelve judges were requested to gather a list of MWEs that normally is seen in contexts. They were asked to collect the data in two languages: English and Portuguese. Furthermore, they were told to label the examples of each MWE (Idiomatic, Non-Idiomatic, Metaphors, Noun). Likewise, the judges were asked to potentially add what the MWE could mean while they gathered the examples. Their main aim was to get a set of three high-quality consecutive sentences with proper grammatical structure. The collected samples were then given to the linguistic experts to be rewrite conserving the true meaning of the sentences for succinctly communicating the idiom's meaning. To exemplify, '*busy bee*' was phrased to '*unavailable bee*' in the literal sense but '*busy* person' in the idiomatic sense also, '*big wig*' was phrased to '*large wig*' literally and '*important person*' idiomatically. This was all introduced so that the model could have some sense about what the word 'idiomaticity' meant for them to triumph. Hence, this data set contains both MWE that is used as a literal meaning and a metaphorical meaning along with surrounding sentences to provide more contextual data.

5.3 Data Processing

As discussed above in Section 4.1, the raw data has eight columns which was divided into 4 splits (training, development, evaluation, testing). For this project, I've used the training set and the testing set. The data is processed using techniques applied in the baseline system of the Task A by using AStitchInLanguageModels (Madabushi, et al., 2021). Hence, there are no missing values and irrelevant features in the datasets.

There are two types of setting provided for the task: zero-shot & one-shot. And there are 3 separate datasets for two different stages: training dataset and testing dataset for both the setting.

I. Zero-shot training data:

In this setting, the target sentence is given along with the previous sentence and next sentence that comes after it along with label and language.

- There is total 4491 rows with 9 columns in this dataset.
- Two rows that has null values were dropped.
- Total of 236 unique MWE exists in the dataset from which 163 are in English and 73 are in Portuguese.

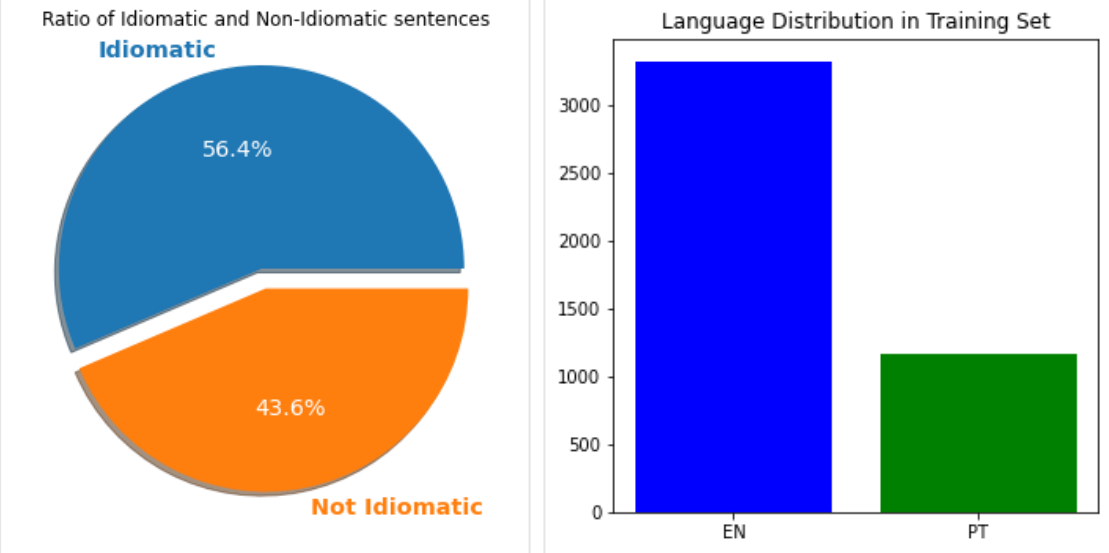

Figure 7: Language & Label classes distribution in zero-shot training data

- About 75% of the rows are in English at 3325 whereas only 25% of it are in Portuguese at 1164.
- For the label class there are 2533 confirmed idiomatic sentences and 1956 non-idiomatic sentences.

Sentence	Label
That's not a fun place for Team Biden to be in. It's going to be a ***blood bath***. Maybe McDonough can swiftly use his crisis management skills to skillfully evade blame for this mess and others Trump has left him.	0
Made In Hollywood TeenVideosMovie TrailersMoviefone's UnscriptedExclusive InterviewsNewsMovie NewsTV NewsPhoto Galleries. Deborah Loomis is an actress, known for Hercules in New York (1970), Foreplay (1975) and ***Blood Bath*** (1976). In 1969, she appeared in three episodes of the ABC Gothic soap opera Dark Shadows where she played a character named Tessie Kincaid.	1

Table 1: Data Structure for Zero-Shot training Data

II. One-Shot training data:

In this setting, the target sentence is passed along with the MWE that the sentence contains, the language and label of the datapoint.

- There is a total of 140 rows with eight columns.
- 87 rows are in English (62%), and 53 rows are in Portuguese (38%).
- 80 non-idiomatic sentences and 60 idiomatic sentences.

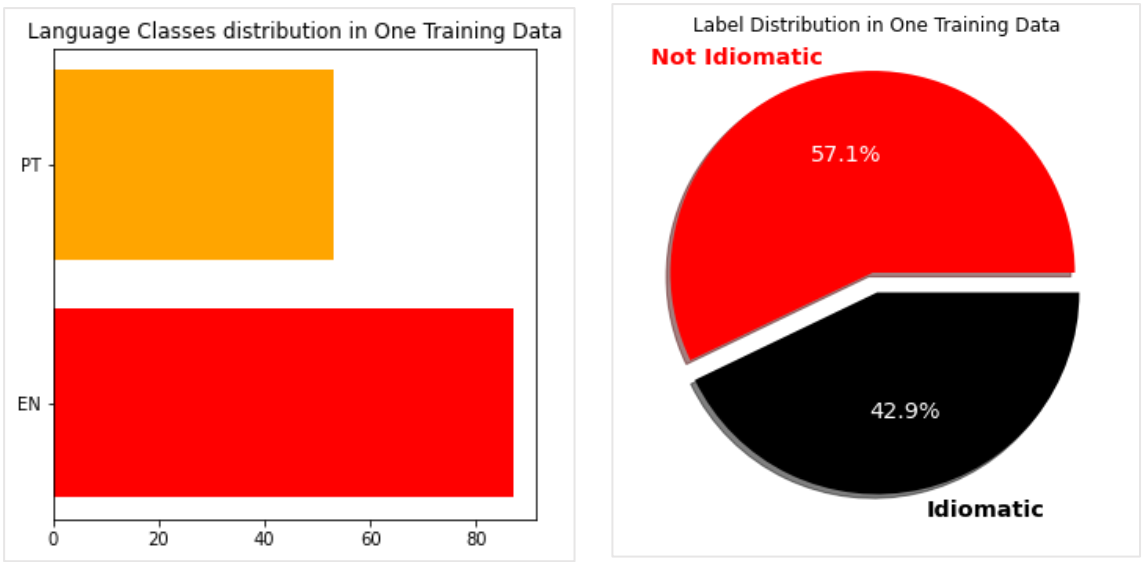

Figure 8: Language & Label classes distribution in one-shot training data

- 100 unique MWEs where 60 of them is in English while 40 of them is in Portuguese.
- 40 MWE has both idiomatic and non-idiomatic samples
- MWE that has only non-idiomatic samples are about 40 and 20 have only idiomatic samples.

MWE	Sentence	Label
blood bath	It's going to be a ***blood bath***.	0
blood bath	Deborah Loomis is an actress, known for Hercules in New York (1970), Foreplay (1975) and ***Blood Bath*** (1976).	1

Table 2: Data Structure for One-shot training data

III. Development Data:

- There is a total of 739 rows with 8 columns.
- Out of 739 rows, 466 are in English (62%) whereas 273 are in Portuguese (38%).
- 403 are non-idiomatic labelled and 336 are idiomatic.
- There are total of 50 unique MWE of which 30 are in English and 20 are in Portuguese.

IV. Comparison between three datasets:

As described in (Madabushi, et al., 2022), there is no overlapping between the zero-shot training dataset and one-shot training dataset neither with dev dataset. Although, all the MWE's that are present in dev are all present in one-shot data.

6. Workflow of the System

a) Data Splitting

The processed and clean data is further split into training and testing set. The training set is used for training the model with ideal parameters to learn from the data. Similarly, testing data is for evaluating the capacity of the model to find patterns in fresh unexplored data. The main purpose of using test dataset is to prevent the model from overfitting, which might result to inefficient or failed model. Generally, the data is split into 80:20 ratio.

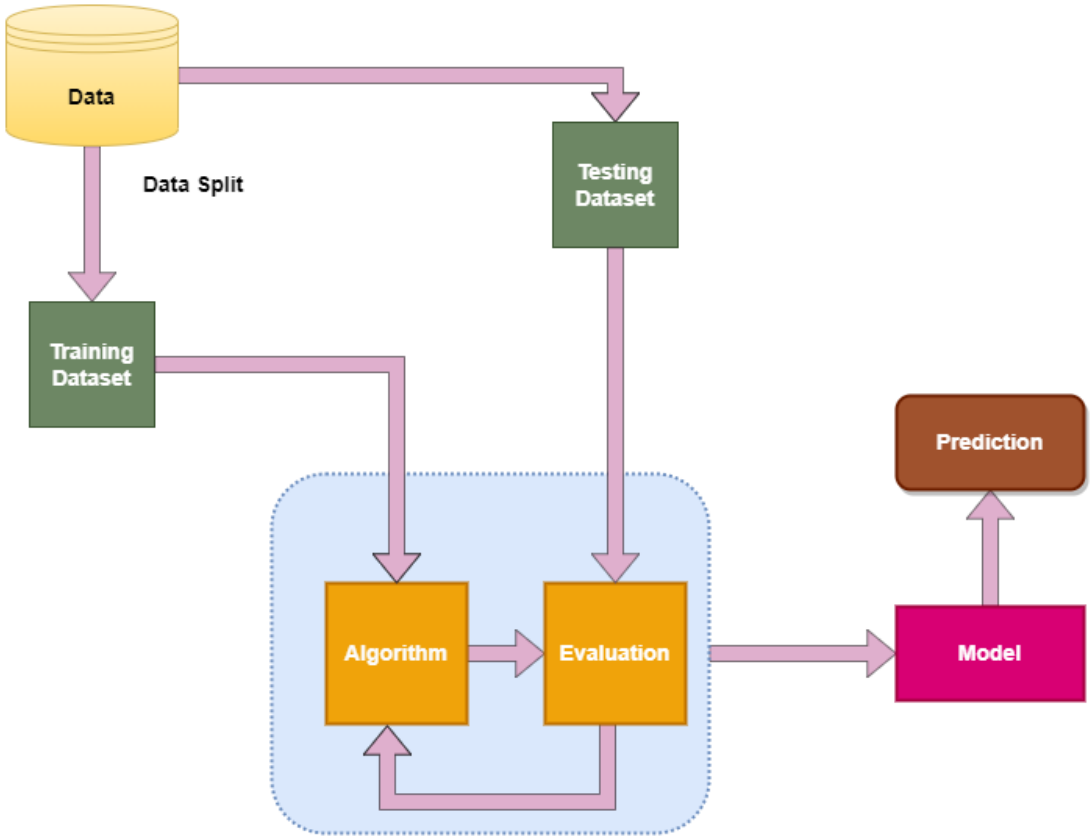

Figure 9: Workflow of the system

b) Modeling

This plays the crucial component in the workflow of the system. It's the phase where the model is engineered by writing and running machine algorithms with the obtained data in previous step. There are quite a few popular methods to apply like **Regression** (ideal for

predicting continuous or numerical values), **Classification** (ideal for classifying between two or more categories) and **Clustering** (ideal for finding certain patterns in the data). For this project, the task is to classify whether a sentence contains idiomatic expression or not.

There are certain steps to create the modeling pipeline:

- **Train the model**

  This is the procedure of using the training dataset to run a machine learning algorithm. There are several available algorithms, thus experimenting with various techniques might be a great way to start training the model. The feature extraction and hyper tuning parameters for a better performing model is also conducted here. Often, the results obtained from the training set is also extracted for it to be compared with the testing set results.

- **Evaluating the model**

  Prior to deployment, it is essential for the model to be evaluated and studied for how it performed so that the target is met. Furthermore, it is crucial step to ascertain whether there are any other aspects that might have been gone unnoticed. The final model must be evaluated using the right metrics in order to ensure that the outputs are efficient. The most popular metric in case of Classification is Accuracy, F1 score, Precision & Recall. In addition, confusion matrix can also be developed to shine a light into how the model is working.

6.1 Metrics

6.1.1 Confusion Matrix

It's a matrix that shows how the classifier is performing based on various experimental data. It can be used to binary or even multiple classes too.

a. **True positives:**

   These are the times that the model predicted were true and matched exactly with the respective output.

b. **True negatives:**

   The model predicted false, but the target was true.

c. **False positives:**

   Here, the model predicted True, but the actual value was False. This is the Type I error.

d. False Negatives:

The model said it was false but correct value was True. This is the Type II error.

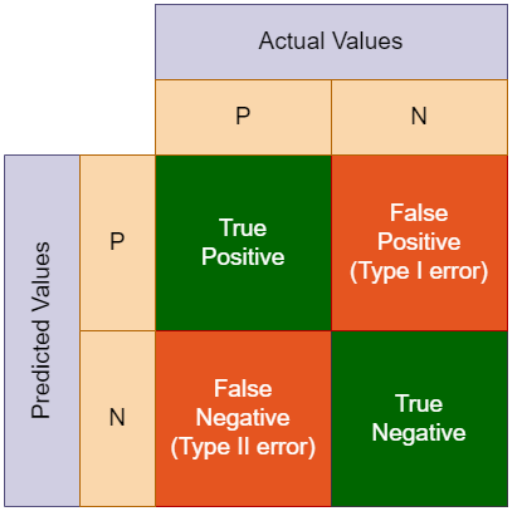

Figure 10: Confusion Matrix Mechanism

6.1.2 Accuracy

One of the parameters for assessing the results of classification models is the accuracy score. It is the percentage of predictions that the model successfully predicted out of all total number of predictions made. Higher the value of accuracy, better the model.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

where TP is the True Positive, FP is the False Positive, FN is the False Negative & TN is the True Negative.

One of the disadvantages of only using Accuracy score to evaluate the model is that it's not really revealing how the model working if only used as the single metric. While, for a classification problem, if the classes are imbalanced, majority classes prediction would be higher compared to the minority ones. Thus, making the model to excel in only predicting a single class.

6.1.3 F1-score

The accuracy of a model on a dataset is measured by the F1-score. It's employed to assess binary categorization schemes that label examples as "positive" or "negative." The harmonic means of the classifier's precision score and recall score is both encapsulated

into the single metric i.e., F1-score (Kumar, 2022). Its primary purpose is to compare two classifiers.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

7. Experiments

The main reason for breaking down the data into two settings to test with is to evaluate how the model performs when the contexts is provided(zero-shot) and when the MWE is provided(one-shot). For both settings, I used pre-trained transformer models: Multilingual BERT, XLM-RoBERTa, XL-NET, Multilingual DistilBERT for comparison. For the optimizer, Adam (Kingma & Ba, 2014) is used. There are three language setting that is tested with.

- English: Only English data are fed to the model.
- Portuguese: Only Portuguese data are fed to the model.
- English Portuguese: Both the language data are fed to the model.

7.1 Hyperparameter tuning

To control the machine learning model's behaviors, certain parameters are passed while training. So, to manipulate the model, these hyperparameters play a huge role. If the model's parameters are tested with multiple values for parameters, then it will aid to minimize the loss function. It operates by conducting multiple trials in a single training procedure. Every trial entails the full execution of the training process using the selected hyperparameters that is set from starting point to ending point. By the end of hyperparameter tuning, the ideal set of hyperparameter values will be selected for the model to perform the best it can.

7.2 Siamese Network

The raw data is processed into creating a dataset where the two sentences with the same MWE are linked in a single datapoint for it to be compared using the Network. The architecture of this network is based on SBERT (Reimers & Gurevych, 2019). This network is built using PyTorch.

7.2.1 Training Procedure

I used XLM-R_{base} as the encoder to use for my Siamese Network. Firstly, the preprocessed dataset is then tokenized for both the sentences. Both the embeddings are then fed to Feed forward network where the output feature vector's element wise difference is taken, this is based on the classification objective function used in SBERT.

The similarity function between two embedded inputs is calculated using:

$$(O_{i,j}) = layer(x1, x2)$$

Where the layer () is the connected layer with sigmoid activation, x1 & x2 are the 2 inputs.

The cross-entropy loss for is calculated using this function:

$$Loss(t, p) = -\sum_{i=1}^{S} (t_i \log(p_i) + (1 - t_i) \log(1 - p_i)$$

Where t is the true label and p is the predicted label with S data samples. The total train loss is calculated for each epoch.

The best parameters for the Siamese model to train in are:

- Learning rate = 2e-5
- Epochs =2
- Optimizer = AdamW
- Batch Size =16
- Maximum Length = 128

7.2.2 Evaluating Procedure

For the testing dataset, three of the sentences with same MWE are grouped together to test the similarities between them. It works like triplet objective function (Reimers & Gurevych, 2019). So, one is the anchor sentence which is then compared with the positive sentence and the negative sentence.

7.3 Ensemble Model

For creating an ensemble mode, the techniques of multiple learning algorithms are combined. Ensemble learning is used to improve the performance of a model when the task a classification, regression etc. When compared to the base learners alone, the model developed performs better. There are three techniques to create an ensemble model: Bagging, Boosting & Stacking.

To combine the results of three transformer models, I've used the Stacking technique.

7.3.1 Stacking Ensemble:

Discovering a place where multiple model's abilities to predict is used, and this is called stacking. The theory is that when approaching a learning problem with different models, it can learn maybe a small portion of it but not the whole.

In the case of the project, I used the predicted labels of the testing set from DistilBERT, XLM-R & XL-NET. Furthermore, the predicted labels are then used as features for the

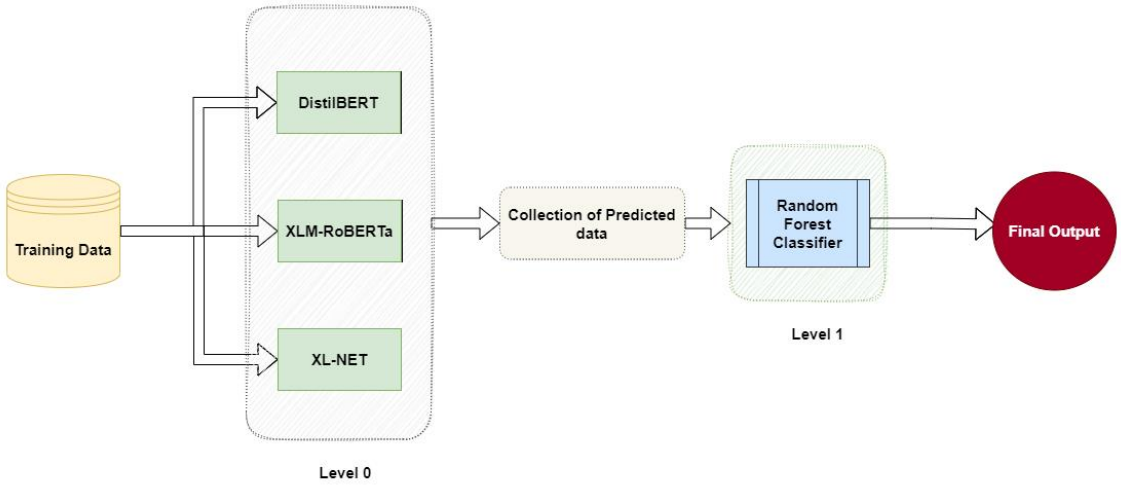

Figure 11: Stacking Ensemble Model

Level 1 model which is the Random Forest Classifier. The model acquired at the end usually performs quite well, mostly exceling compared to the individual Level 0 models in terms of performance.

7.4 Resampling

Resampling refers to the general process of creating new samples from an existing dataset. In the context of imbalanced datasets, resampling can be used to create a balanced dataset by either removing observations from the majority class (undersampling) or by adding copies of observations from the minority class (oversampling).

7.4.1 Undersampling

Undersampling is a technique in which the majority class is reduced to the size of the minority class by randomly removing observations from the majority class. This is done to balance the dataset, but it can lead to a loss of information.

7.4.2 Oversampling

Oversampling, on the other hand, is a technique in which additional copies of the minority class are added to the dataset. This can be done by randomly replicating observations from the minority class or by generating synthetic samples. Oversampling can increase the number of examples in the minority class but it can also lead to overfitting.

Both oversampling and undersampling have their own advantages and disadvantages, and the choice of which to use will depend on the specific problem and dataset being used. A combination of both oversampling and undersampling can also be used to improve the performance of the model.

8. Results

8.1 Zero-shot setting

8.1.1 Comparison results of Transformer models

As stated above in Section 7, I experimented with three transformer models i.e., DistilBERT, XLM-RoBERTa & XL-NET.

Language	Model	Accuracy	F1-Score	Best Model
EN	mBERT	0.70	0.67	**XLM-R_{large}**
	XLM-R_{base}	0.72	0.71	
	XLM-R_{large}	**0.77**	**0.75**	
	DistilBERT	0.69	0.68	
	XL-NET	0.75	0.74	
PT	mBERT	0.67	0.66	**XLM-R_{base}**
	XLM-R_{base}	**0.68**	**0.67**	
	XLM-R_{large}	0.67	0.67	
	DistilBERT	0.62	0.60	
	XL-NET	0.58	0.52	
EN-PT	mBERT	0.69	0.68	**XLM-R_{large}**
	XLM-R_{base}	0.71	0.71	
	XLM-R_{large}	**0.73**	**0.72**	
	DistilBERT	0.67	0.67	
	XL-NET	0.69	0.69	

Table 3: Comparison of different Transformer models for zero-shot setting

Table 3 shows the Accuracy and F1 score obtained for different language settings for various transformer models along with the baseline(mBERT) in zero-shot setting. It can be observed that XLM-R_{large} performed the best among the rest with Accuracy (0.77) & F1-score (0.75) for English only, also with Accuracy (0.73) & F1-score (0.72) while feeding both English & Portuguese at the same time. Although, XLM-R_{base} worked best for Portuguese with Accuracy (0.68) & F1-score (0.67).

8.1.2 Resampling Language data

Language	Accuracy	F1-Score
EN	**0.74**	**0.74**
PT	0.62	0.59
EN-PT	0.70	0.70

Table 4: Performance of a balanced language dataset

Biased results can be obtained if the binary classes or multiple classes of the data are not equal all-in number. Hence, for getting a fair result in terms of Language class, I've balanced the number of English data & Portuguese data for training set. I used the best performing model XLM-RoBERTa$_{large}$ for this experiment.

The results given by balanced model is promising when it comes to English language but for the Portuguese language, its in the downward direction. The confusion matrix in Figure

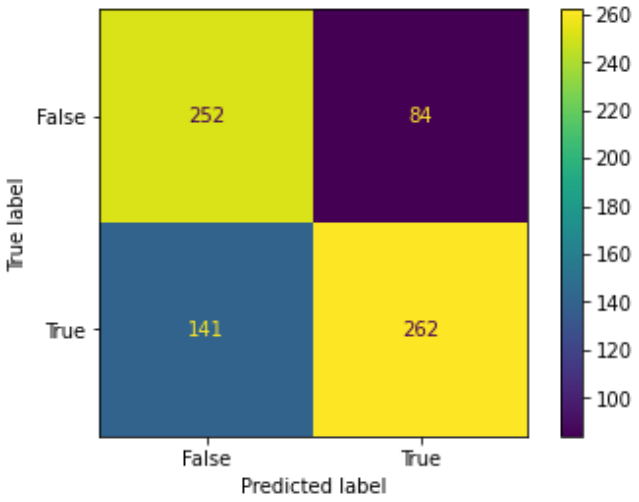

Figure 12: Confusion Matrix of language balanced zero-shot model

12, shows that the number of correctly predicted classes is quite high overall but the Type I error(False Positive) is not ignorable.

8.1.3 Ensemble Model

Language	**Accuracy**	**F1-Score**
EN	0.75	**0.75**
PT	0.67	**0.67**
EN-PT	0.67	0.67

Table 5: Ensemble Model's performance

By collecting the predicted labels of testing data that models XLM-R_{large}, XL-NET, DistilBERT gave, I created an ensemble model which used the stacking technique as mentioned in Section 7.3.1 by passing the labels as features against the true labels as target. The dataset is also resampled according to the target classes (0 for idiomatic & 1 for non-idiomatic) for an unbiased result. Furthermore, data is spilt into training and

testing set with the ratio of 80:20 for it to be passed into the classifier. According to Table 4, the ensemble model didn't outperform the best model (XLM-R), the F1-score for both EN & PT were tied with it. But when compared with the baseline model (mBERT), for English language, the ensemble model outperformed it.

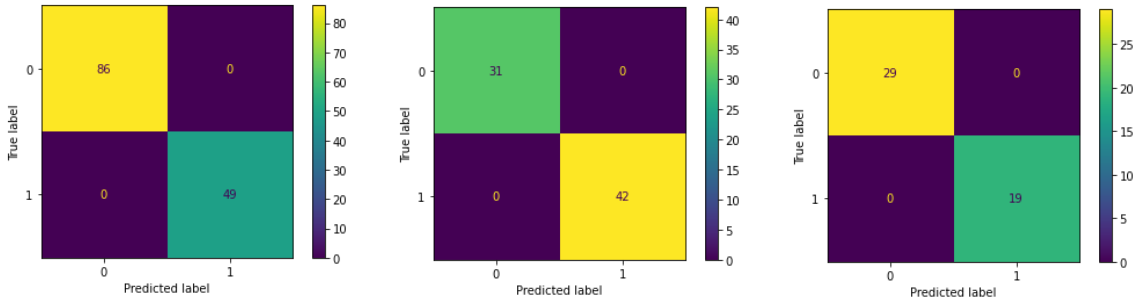

Figure 13: Confusion Matrix for EN-PT, EN, PT

From Figure 13, the confusion matrix of all the language settings depicts that the models correctly predicted the right classes and there was no Type I error (FP) or Type II error (FN).

8.2 One-shot setting

8.2.1 Comparison results of Transformer Models

Language	Model	Accuracy	F1-Score	Best Model
EN	mBERT	0.87	0.86	**$XLM\text{-}R_{large}$**
	$XLM\text{-}R_{base}$	0.89	0.89	
	$XLM\text{-}R_{large}$	**0.92**	**0.91**	
	DistilBERT	0.83	0.82	
	XL-NET	0.90	0.89	
PT	mBERT	0.88	0.87	**$XLM\text{-}R_{large}$**
	$XLM\text{-}R_{base}$	0.88	0.87	
	$XLM\text{-}R_{large}$	**0.89**	**0.89**	

	DistilBERT	0.82	0.81	
	XL-NET	0.80	0.79	
EN-PT	mBERT	0.87	0.87	**XLM-R_{large}**
	XLM-R_{base}	0.88	0.88	
	XLM-R_{large}	**0.90**	**0.90**	
	DistilBERT	0.83	0.83	
	XL-NET	0.86	0.86	

Table 6: Comparison of different Transformer models for one-shot setting

Table 6 shows the performance of each transformer model in terms of Accuracy and F1 score. Overall, XLM-RoBERTa$_{large}$ proved to be the best transformer model for detecting idioms for the one-shot setting. It scored Accuracy (0.92, 0.89, 0.90) & F1(0.91,0.89,0.90) for the English, Portuguese & English-Portuguese language settings.

8.2.2 Resampling Language Data

The resampling of the Language categories both are sampled to the equal number exactly like in the zero-shot setting. XLM-RoBERTa$_{large}$ was used for this experiment.

The evaluation metrics for this setting is declined immensely compared with the unbalanced setting (Table 6).

Language	**Accuracy**	**F1-Score**
EN	**0.68**	**0.66**
PT	0.64	0.60
EN-PT	0.66	0.66

Table 7: Performance of Language Balanced dataset for one-shot setting

From Figure 14, by comparing the two matrices, unbalanced data's model performed better compared to the balanced one as the Type II error (False Negative) is quite high.

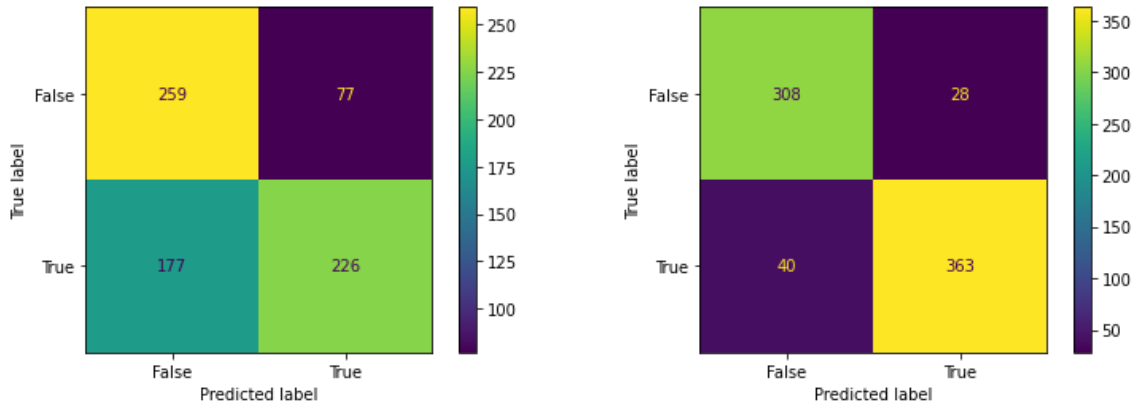

Figure 14: Confusion Matrix for Balanced & Unbalanced Dataset for one-shot setting

8.2.3 Ensemble Model

Language	Accuracy	F1-Score
EN	0.90	0.90
PT	**0.92**	**0.92**
EN-PT	**0.91**	**0.91**

Table 8: Ensemble Model's performance for one-shot setting

This model was created as shown in Figure 11, by combining the results of three transformer models using the stacking technique. For a fair outcome, the dataset is also resampled according to the desired classes (idiomatic and non-idiomatic). Additionally, data is split into training and testing sets in an 80:20 ratio before being fed into the classifier.

According to Table 8, the ensemble model outscored the XLM-RoBERTa (highest performing model for one-shot) in the case of Portuguese with a 3% improvement in both scores.

Similarly, the English Portuguese saw a 1% growth. However, the confusion matrix suggests that this may not be a desirable change. Even if the scores are good, the False Positives (Type I Error) appears to be the maximum, which indicates that the model is not performing efficiently.

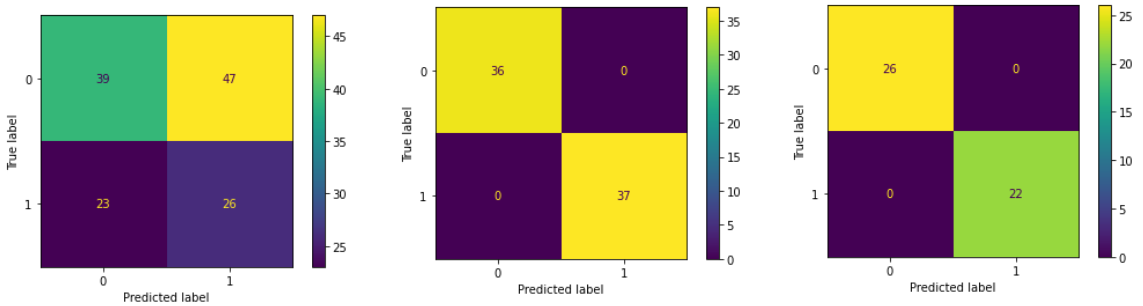

Figure 15: Confusion Matrix for EN-PT, EN, PT for one-shot

8.2.4 Siamese Network

I implemented Siamese Network to the best performing transformer model which is XLM-RoBERTa$_{base}$ as per the results obtained in Section 8.2.1 The metrics that I used to evaluate these models are the accuracy and the F1 score.

Language	Accuracy	F1-Score
EN	0.86	0.86
PT	0.84	0.84
EN-PT	0.85	0.85

Table 9: Evaluation Metrics of Siamese Network implemented XLM-R model

As seen in Table 9, the Siamese Network implemented XLM-R didn't outperform the original model. Even the confusion matrix for all the language settings seems promising as both the errors is less.

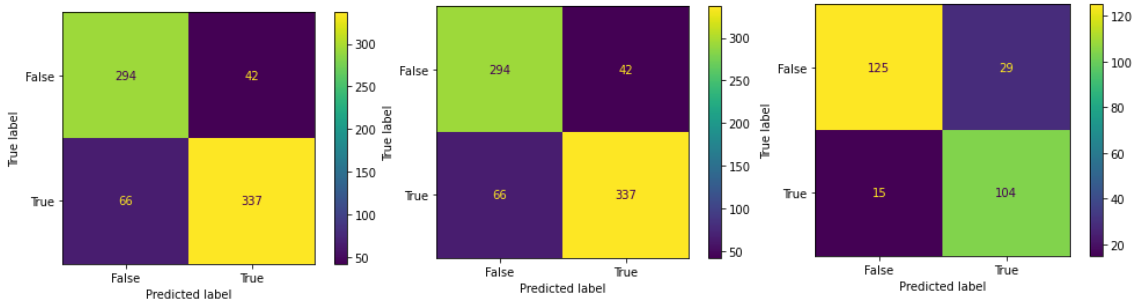

Figure 16: Confusion Matrix for EN-PT, EN, PT for one-shot

9. Conclusion

9.1 Overview

To sum up, the main goal of this project was to detect idioms in natural language. Broad literature regarding how the existing systems detect idioms were discussed which were highly and mildly related to this topic. Transformer models have shown quite a remarkable result as compared to the neural networks like Recurrent Neural Network (RNN) or long short-term memory (LSTM). The attention mechanism enables the model to concentrate on the input during processing is one of the main reasons why Transformer models excel. There are lots of dependencies when it comes to textual data which is very frequent in human languages. Transformer models can accommodate input-sequences of various lengths which is crucial while dealing with textual data of different lengths which is made possible by the self-mechanism technique. Models like BERT, GPT-3 have been pre-trained on a substantial amount of textual data which is why they are able to comprehend and produce texts like humans for variety of task. Furthermore, the same architecture can be used for multiple tasks in case of transformers with a bit of fine-tuning although RNNs aren't that flexible.

Sentences from two languages were used: English and Portuguese for training the model.

9.2 Findings

Idiomatic expressions can be quite rarely used and is dynamic overtime. Furthermore, new idioms might show up in a small span of time. They are very language specific and cannot quite sound right when translated. Hence, there are less data available for existing idioms in any languages. The dataset that I used roughly has about 4000 data points which is not quite enough to train a model due to the complexity of this task and the diversity in sentences for each Multiword expression. In the dataset, it's not certain that every MWE has sample for both idiomatic and non-idiomatic which is essential.

I suspected that models were able to predict English idioms better than the Portuguese ones when I trained the model regardless of their quantities which was 63:37 ratio as seen in Section 8.2.1 and 8.1.1. English has better scores than Portuguese in both settings. So, does the fine-tuned transformer models really work better for English in this specific task? And the answer was YES. The multilingual transformer-based model does work better in

English even after the Language class is balanced for both the zero-shot and one-shot setting (Section 8.2.2 and 8.2.1). One of the models I used was mBERT which was trained in 104 languages using data from Wikipedia and webpages from Common Crawl. There are approximately 25,000,000 documents and 174,000,000 tokens in the corpus. Since English is one of the most frequently spoken languages in the world, it is likely to have the most data of any language in the corpus. Since, the texts in the corpus were not labelled with the languages they were written in, mBERT did not have a predefined amount of data for each language. Furthermore, experimenting with the two settings showed than the models perform reasonably well for one-shot scenarios while the models still struggle to detect never seen MWEs when testing with the zero-shot setting.

Ultimately, using Siamese Network didn't outperform the baseline or the best working transformer model. The

9.3 Future Work

9.3.1 Unsupervised learning

This project is focused on a supervised machine learning technique that works on labeled data for two languages. Rather than approaching this task as a binary classification, I would utilize unsupervised machine learning technique like clustering to identify the idioms based on the data collocations and their occurrence patterns.

9.3.2 Focusing on a Single Language Model

Idiomaticity detection for many languages can be handled by a multilingual model trained on a broad set of data from different languages, although it might not be as effective on any particular language as a single language model would. This is because the model will not have had as much exposure to data from any particular language and may not have learnt the structures and patterns of idiomatic expressions in any particular language as well as a model trained on only one language would. Hence, focusing on using a language specific pre-trained model rather than multilingual transformer model would be something I would try. A single language transformer model will probably do better at idiomaticity detection for a given language than a multilingual model if it has been trained on a large corpus of data from that language. This is because the model will have encountered many idiomatic expressions unique to that language and will have become familiar with their

patterns and structures. The model will also be trained to comprehend the linguistic and cultural context of idioms used in that language.

9.3.3 Possibilities in other Language

Future research should focus on idiom detection in underrepresented languages since it can assist NLP systems perform better in these languages and give users of those languages more access to language technology. For the future note, I intend to proceed by collecting data from around the social media and creating a survey using Google Forms survey for the native speakers for underrepresented South Asian Languages like Hindi, Nepali etc.

References

Baldwin, T. & Kim, S. N., 2010. Multiword expressions.. *Handbook of natural language processing,* Volume 2, pp. 267-292.

Barnaby, J., 2022. *Weird and Interesting Nicknames for London… and The Stories Behind Them.* [Online] Available at: https://www.londonxlondon.com/nicknames-for-london/ [Accessed 02 06 2022].

Bromley, J. et al., 1993. Signature Verification using a "Siamese" Time Delay Neural Network. *Advances in Neural Information Processing Systems,* Volume 6.

Conneau, A. et al., 2019. *Unsupervised Cross-lingual Representation Learning at Scale.* s.l., arXiv.

Constant, M. et al., 2017. Multiword Expression Processing: A Survey. *Computational Linguistics,* 43(0891-2017), pp. 837-892.

Dai, Z. et al., 2019. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context.* s.l., arXiv.

Devlin, J. & Chang, M.-W., 2018. *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing.* [Online] Available at: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html [Accessed 30 12 2022].

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K., 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* Minnesota, Association for Computational Linguistics.

Gross, M., 1982. Une classification des phrases figées du français. *Revue québécoise de linguistique,* 11(2), pp. 151-185.

Haagsma, H., Bos, J. & Nissim, M., 2020. *MAGPIE: A Large Corpus of Potentially Idiomatic Expressions.* Marseille, European Language Resources Association.

Hashempour, R. & Villavicencio, A., 2020. Leveraging Contextual Embeddings and Idiom Principle for Detecting Idiomaticity in Potentially Idiomatic Expressions. *Proceedings of the Workshop on the Cognitive Aspects of the Lexicon,* pp. 72-80.

Hinton, G., Vinyals, O. & Dean, J., 2015. *Distilling the Knowledge in a Neural Network.* s.l., arXiv.

Hovy, D. & Spruit, S. L., 2016. The social impact of natural language processing. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers).* s.l.:s.n., pp. 591-598.

Kingma, D. P. & Ba, J., 2014. *Adam: A Method for Stochastic Optimization.* s.l., arXiv.

Koch, G., Zemel, R. & Salakhutdinov, R., 2015. Siamese neural networks for one-shot image recognition. *ICML deep learning workshop,* Volume 2, p. 0.

Kumar, A., 2022. *Accuracy, Precision, Recall & F1-Score – Python Examples.* [Online] Available at: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/ [Accessed 3 1 2023].

Legrand, J. & Collobert, R., 2016. *Phrase representations for multiword expressions.* s.l., s.n.

Libovický, J., Rosa, R. & Fraser, A., 2019. *How Language-Neutral is Multilingual BERT?.* Prague, arXiv.

Liu, Y. et al., 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach.* s.l., arXiv.

Madabushi, H. T. et al., 2022. *SemEval-2022 Task 2: Multilingual Idiomaticity Detection and Sentence Embedding.* s.l., arXiv.

Madabushi, H. T., Gow-Smith, E., Scarton, C. & Villavicencio, A., 2021. AStitchInLanguageModels: Dataset and Methods for the Exploration of Idiomaticity in Pre-Trained Language Models. *CoRR,* Volume abs/2109.04413.

Martin, L. et al., 2020. *CamemBERT: a Tasty French Language Model.* Paris, Association for Computational Linguistics.

Melamud, O., Goldberger, J. & Dagan, I., 2016. context2vec: Learning generic context embedding with bidirectional lstm. *Proceedings of the 20th SIGNLL conference on computational natural language learning,* pp. 51-61.

Meta AI, 2019. *RoBERTa: An optimized method for pretraining self-supervised NLP systems.* [Online]
Available at: https://ai.facebook.com/blog/roberta-an-optimized-method-for-pretraining-

self-supervised-nlp-systems/
[Accessed 15 12 2022].

Mikolov, T., Chen, K., Corrado, G. & Dean, J., 2013. *Efficient Estimation of Word Representations in Vector Space.* California, arXiv.

Nandakumar, N., Baldwin, T. & Salehi, B., 2019. How Well Do Embedding Models Capture Non-compositionality? A View from Multiword Expressions. *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pp. 27-34.

Phelps, D. et al., 2022. *Sample Efficient Approaches for Idiomaticity Detection.* s.l., arXiv.

Plaza-del-Arco, F. M. M.-V. M. T. U.-L. L. A. M. R., 2020. Improved emotion recognition in Spanish social media through incorporation of lexical knowledge. *Future Generation Computer Systems*, Volume 110, pp. 1000-1008.

Rei, M., Bulat, L., Kiela, D. & Shutova, E., 2017. *Grasping the Finer Point: A Supervised Similarity Network for Metaphor Detection.* s.l., arXiv.

Reimers, N. & Gurevych, I., 2019. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.* s.l., arXiv.

Salehi, B., Cook, P. & Baldwin, T., 2014. Using distributional similarity of multi-way translations to predict multiword expression compositionality. In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics.* s.l.:s.n., pp. 472-481.

Sanh, V., Debut, L., Chaumond, J. & Wolf, T., 2019. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.* s.l., arXiv.

Savary, A. et al., 2017. The PARSEME shared task on automatic identification of verbal multiword expressions. In: *The 13th Workshop on Multiword Expression at EACL.* s.l.:s.n., pp. 31-47.

Schick, T. & Schütze, H., 2019. *BERTRAM: Improved Word Embeddings Have Big Impact on Contextualized Model Performance.* Munich, Germany, arXiv.

Sinclair, J. & Sinclair, L., 1991. *Corpus, concordance, collocation.* s.l.:Oxford University Press.

Vaswani, A. et al., 2017. *Attention Is All You Need.* s.l., arXiv.

Verma, Y., 2022. *A complete tutorial on masked language modelling using BERT.* [Online]
Available at: https://analyticsindiamag.com/a-complete-tutorial-on-masked-language-modelling-using-bert/
[Accessed 16 10 2022].

Yang, Z. et al., 2019. *XLNet: Generalized Autoregressive Pretraining for Language Understanding.* s.l., arXiv.

Zeng, Z. & Bhat, S., 2021. Idiomatic Expression Identification using Semantic Compatibility. *Transactions of the Association for Computational Linguistics*, 9(2307-387X), pp. 1546-1562.

Appendix

For this project, I used Google Colab Pro which provides more memory and faster CPU, longer runtimes, more storage etc. I used the transformer models which are large neural networks that require significant computational resources to train and run. The transformer architecture is composed of multiple layers of self-attention, which are computationally expensive to compute. Because of this, training and running transformer models on a CPU can be slow and impractical. Although, GPUs (Graphics Processing Units) are specialized hardware devices that are designed to accelerate the computation of large matrix operations, which are the building blocks of neural networks. GPUs have many more cores than a CPU, which allows them to perform many calculations in parallel, resulting in much faster computation. This makes them well-suited to handle the large amounts of matrix operations required by transformer models. In addition, many deep learning frameworks such as Tensorflow, Pytorch and others have implemented CUDA support, which allows the code to be executed on the GPU, making it easier to train and run transformer models on a GPU.

Creating Siamese Network

This code defines a class called "SiameseModel" which is a PyTorch neural network model. The class inherits from the PyTorch "nn.Module" class, which provides a lot of built-in functionality for creating neural networks. The class has an "init" method that gets called when a new instance of the class is created. This method sets up the architecture of the model. The first line of the init method calls the parent class's constructor, which is necessary for the class to function correctly. It then loads a pre-trained transformer model called "xlm-roberta-base" using the AutoModel.from_pretrained() function from Hugging Face's transformers library. The model is loaded on the GPU by calling cuda(). Then it creates a dropout layer with a dropout rate of 0.5 and a linear layer with 2 output neurons. The input to the linear layer is the dot product of the output of the two inputs. The forward method is called when the model is used to make predictions. It takes in 4 arguments: input_1_id, attention_1_mask, input_2_id, attention_2_mask. It passes these inputs through the pre-trained transformer model and calculates the dot product of the output of

the two inputs. The dot product is passed through the dropout layer and linear layer to get the output.

```
[ ] import torch.nn as nn
    class SiameseModel(nn.Module):
        def __init__(self):
            super(SiameseModel, self).__init__()

            self.base_model = AutoModel.from_pretrained(
                'xlm-roberta-base',
                from_tf=bool(".ckpt" in 'xlm-roberta-base'),
                config=config,
                cache_dir=None,
                revision="main",
                use_auth_token=None,
            ).cuda()
            self.dropout = nn.Dropout(0.5)
            self.linear = nn.Linear(768, 2).cuda()

        def forward(self, input_1_id, attention_1_mask, input_2_id, attention_2_mask):

            first_output = self.base_model(input_1_id, attention_mask=attention_1_mask).last_hidden_state[:, 0]
            second_output = self.base_model(input_2_id,attention_mask=attention_2_mask).last_hidden_state[:, 0]

            outputs = self.dropout(first_output*second_output) #similarity score (dot product)
            outputs = self.linear(outputs)

            return outputs
```

Figure 17: Creating a Siamese network

Training

The AdamW optimizer and a customized train loop are being used in this code to train a model. The model, train data, learning rate, number of epochs, tokenizer, batch size, and maximum length are all inputs to the function. It builds a dataloader for use in a training loop and tokenizes the training data. It runs through the dataloader for each epoch, zeroing gradients at each step, running the model forward, figuring out the cross-entropy loss between the output and the target, and then running backpropagation and optimization steps. The training loss per period is then printed.

```
optimizer = AdamW(model.parameters(),
                  lr=learning_rate,
                  no_deprecation_warning=True)
model.train()

#tokenizing training data
train_dataloader = Preprocess_Data(train_data, tokenizer, maximum_length, batch_size)

# TRAIN loop
for epoch in range(number_of_epochs):
    print(f"\nEpoch {epoch}")
    torch.cuda.empty_cache()
    trainloss = 0
    training_samples, training_steps = 0, 0
    for step, batch in enumerate(tqdm(train_dataloader)):
        batch = tuple(t.to(device) for t in batch)
        model.zero_grad()
        # forward pass
        logits = model.forward(batch[0], batch[1], batch[3], batch[4])
        loss = 0
        # if any of the sentence is idiomatic, the the combined target would be idiomatic
        target = torch.where(batch[2]==batch[5], 1, 0)

        #computes the cross entropy loss between input logits and target.
        loss = nn.functional.cross_entropy(logits, target)

        #Computes the gradient of current tensor
        loss.backward()
        # For tracking the loss for each batch
        trainloss += loss.item() # item() returns the value of this tensor as int number
        training_steps += 1

        # updates parameters
        nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        #does a single optimization step
        optimizer.step()

    # print train loss per epoch
    print("Train loss on epoch {}: {}\n".format(epoch, trainloss / training_steps))
```

Figure 18: Training Procedure of Siamese Model

Evaluating

Utilizing a dataloader, this method evaluates a transformer model. The dataloader is iterated through, and for each batch, the output of the forward pass with the anchor sentence and the positive and negative sentences is applied with the SoftMax function.

The resulting tensors are concatenated, and the index of the maximum value in the concatenated tensor is found and added to a list of predicted labels.

For each anticipated and real label for English and Portuguese sentences, it also stores the real labels. It also keeps track of the quantity of accurate predictions. The accuracy and f1-score are also returned, along with the true and predicted labels for all sentences, English sentences, and Portuguese sentences.

```
def Eval(transformer_model, dataloader):

    transformer_model.eval()
    predictions, true_labels,predictions_en, true_labels_en,predictions_pt, true_labels_pt = ([] for i in range(6))

    for step, batch in enumerate(tqdm(dataloader)):
        batch = tuple(t.to(device) for t in batch)

        #batch[0] = sentence1, batch[1] = attention1, batch[2] =labels1
        #batch[3]= sentence2, batch[4] = attention2, batch[5] =labels2
        #batch[6]= sentence2, batch[7] = attention3, batch[8] =labels3
        #batch[9] = language
        with torch.no_grad():
            #applies softmax function between anchor sentence & positive sentence & returns a tensor
            first_tensor = nn.functional.softmax(transformer_model.forward(batch[0], batch[1], batch[3], batch[4]), -1)
            #applies softmax function between anchor sentence & negative sentence & returns a tensor
            second_tensor = nn.functional.softmax(transformer_model.forward(batch[0], batch[1], batch[6], batch[7]), -1)

        #concatenates two tensors to desired dimension
        logits = torch.cat((first_tensor, second_tensor), dim=1)

        #returns the index where maximum value exists
        max_args = torch.argmax(logits, dim=1)

        batch_both_real_label,first_sentence_labels,second_sentence_labels,LN = batch[2], batch[5], batch[8],batch[9]
        batch_both_predicted_label, batch_en_predicted_label,batch_pt_predicted_label, batch_en_real_label, batch_pt_real_label,sentence_individual = ([] for i in range(6))

        for idx, instance in enumerate(max_args):
          instance_individual = instance.item()
          mapping = {0: lambda x: (x - 1) * -1, 1: lambda x: x, 2: lambda x: (x - 1) * -1, 3: lambda x: x}

          batch_both_predicted_label.append(mapping[instance_individual](first_sentence_labels[idx] if instance_individual < 2 else second_sentence_labels[idx]))
          if LN[idx].item() == 0:
            batch_en_predicted_label.append(mapping[instance_individual](first_sentence_labels[idx] if instance_individual < 2 else second_sentence_labels[idx]))
            batch_en_real_label.append(batch_both_real_label[idx])
          else:
            batch_pt_predicted_label.append(mapping[instance_individual](first_sentence_labels[idx] if instance_individual < 2 else second_sentence_labels[idx]))
            batch_pt_real_label.append(batch_both_real_label[idx])

        predictions += batch_both_predicted_label
        true_labels += batch_both_real_label
        predictions_en += batch_en_predicted_label
        predictions_pt += batch_pt_predicted_label
        true_labels_en += batch_en_real_label
        true_labels_pt += batch_pt_real_label
    return true_labels, predictions, true_labels_en, predictions_en, true_labels_pt, predictions_pt
```

Figure 19: Evaluating procedure of Siamese Model

Codes Citations

Some codes from the baseline system were used while developing the project. GitHub: https://github.com/h-tayyarmadabushi/semeval_2022_task2-idiomaticity