UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GABRIEL COUTO DOMINGUES

Evaluating data imbalance approaches for classifying semantic relations using machine learning and word embeddings

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Joel Luís Carbonera
Coadvisor: MSc. Alcides Gonçalves Lopes Junior

Porto Alegre
September 2023

ABSTRACT

Explicit knowledge models are artifacts that represent domain knowledge in an explicit way and can be used in different ways, including structuring data, supporting information retrieval and reasoning. The identification and classification of semantic relationships between concepts is a critical task in the development of knowledge models. This work investigates the use of machine learning approaches and pre-trained static word embeddings to classify semantic relationships between concepts, evaluating different techniques to deal with the challenges imposed by data imbalance in this context. We proposed a methodology for building datasets for the task of semantic relationship classification from word embeddings using WordNet as a semantic reference. By applying the proposed methodology, we generated two different datasets, with two variations, for the target task. Finally, we evaluated a set of general approaches for dealing with data imbalance in classification tasks. Our results indicated that while some strategies like SMOTE showed promise in specific metrics, the baseline model consistently achieved superior performance in terms of F1 score.

Keywords: Word Embeddings. Machine Learning. Supervised Learning. Neural Networks. Ontologies. Knowledge Graphs. WordNet. Semantic Relationships.

RESUMO

Modelos de conhecimento explícito são artefatos que representam conhecimento de domínio de forma explícita e podem ser usados de diferentes maneiras, incluindo estruturação de dados e suporte à recuperação de informações e raciocínio. A identificação e classificação das relações semânticas entre conceitos é uma tarefa crítica no desenvolvimento de modelos de conhecimento. Este trabalho investiga o uso de abordagens de aprendizado de máquina e *word embeddings* estáticos pré-treinados para classificar relações semânticas entre conceitos, avaliando diferentes técnicas para lidar com os desafios impostos por dados desbalanceados neste contexto. Propomos uma metodologia para construir conjuntos de dados para a tarefa de classificação de relações semânticas a partir de *word embeddings* usando o WordNet como referência semântica. Ao aplicar a metodologia proposta, geramos dois conjuntos de dados diferentes, com duas variações, para a tarefa de classificação. Por fim, avaliamos um conjunto de abordagens gerais para lidar com desbalanceamento de dados em tarefas de classificação. Nossos resultados indicaram que, enquanto algumas estratégias, como o SMOTE, mostraram promessa em métricas específicas, o modelo base demonstrou consistentemente um desempenho superior em termos de *F1 score*.

Palavras-chave: *Word Embeddings*, Aprendizado de Máquina, Aprendizado Supervisionado, Redes Neurais, Ontologias, Grafos de Conhecimento, WordNet, Relações Semânticas.

LIST OF FIGURES

Figure 2.1 Representation of neural networks. 14
Figure 2.2 Confusion matrix with evaluation metrics. 17
Figure 2.3 SMOTE-Tomek: (a) original data set ; (b) over-sampled data set; (c) Tomek links identification; and (d) borderline and noise examples removed. 21
Figure 2.4 Borderline SMOTE: (a) The original distribution of an example dataset. (b) The borderline minority examples (solid squares). (c) The borderline synthetic minority examples (hollow squares). 21
Figure 2.5 Example of an ontology. 23
Figure 2.6 Example of a knowledge graph for flight data. 24
Figure 2.7 Network representation of hyponymy, antonymy and meronymy relations among an illustrative variety of lexical concepts. 26

Figure 4.1 Example of different meanings of the word "car". 33
Figure 4.2 Processing pipeline. 34
Figure 4.3 Difference in number of matches with both datasets with and without the matching and normalization rules. 36
Figure 4.4 Percentage of pairs of words of each relationship per dataset: A) WordNet, B) Gensim, C) FastText. 37
Figure 4.5 Neural network model used for the classification task. 39

Figure 5.1 Average confusion matrices for baseline model with three relationships. 42
Figure 5.2 Average confusion matrices for weighted loss experiments with three relationships. 44
Figure 5.3 Average confusion matrices for focal loss experiments with three relationships. 44
Figure 5.4 Average confusion matrices for random under-sampling experiments with three relationships. 46
Figure 5.5 Average confusion matrices for doubled random under-sampling experiments with three relationships. 46
Figure 5.6 Average confusion matrices for Tomek links experiments with three relationships. 47
Figure 5.7 Average confusion matrices for SMOTE experiments with three relationships. 48
Figure 5.8 Average confusion matrices for SMOTE/Tomek experiments with three relationships. 49
Figure 5.9 Average confusion matrices for Borderline SMOTE experiments with three relationships. 49
Figure 5.10 Average confusion matrices for baseline model with two relationships. 52
Figure 5.11 Average confusion matrices for weighted loss experiments with two relationships. 53
Figure 5.12 Average confusion matrices for focal loss experiments with two relationships. 54
Figure 5.13 Average confusion matrices for random under-sampling experiments with two relationships. 55
Figure 5.14 Average confusion matrices for doubled random under-sampling experiments with two relationships. 56
Figure 5.15 Average confusion matrices for Tomek links experiments with two relationships. 56

Figure 5.16 Average confusion matrices for SMOTE experiments with two relationships. 58
Figure 5.17 Average confusion matrices for SMOTE/Tomek experiments with two relationships. 58
Figure 5.18 Average confusion matrices for Borderline SMOTE experiments with two relationships. 59
Figure 5.19 Macro F1 scores from the experiments with three relationships and two relationships for Gensim. 60
Figure 5.20 Macro F1 scores from the experiments with three relationships and two relationships for FastText. 60

LIST OF TABLES

Table 4.1 Number of pairs of words of each relationship per dataset. 36
Table 4.2 Percentage of pairs of words of each relationship per dataset. 37
Table 4.3 Number of Pairs of words with more than one relationship per dataset. 37
Table 4.4 Percentage of pairs of words with more than one relationship per dataset. 38

Table 5.1 Evaluation metrics for the baseline model with three relationships. 42
Table 5.2 Evaluation metrics for loss changes for Gensim dataset with three relationships. 43
Table 5.3 Evaluation metrics for loss changes for FastText dataset with three relationships. 43
Table 5.4 Evaluation metrics for under-sampling for Gensim dataset with three relationships. 45
Table 5.5 Evaluation metrics for under-sampling for FastText dataset with three relationships. 45
Table 5.6 Evaluation metrics for over-sampling for Gensim dataset with three relationships. 47
Table 5.7 Evaluation metrics for over-sampling for FastText dataset with three relationships. 48
Table 5.8 Evaluation metrics for experiments for Gensim dataset with three relationships. 50
Table 5.9 Evaluation metrics for experiments for FastText dataset with three relationships. 50
Table 5.10 Evaluation metrics for baseline model with two relationships. 51
Table 5.11 Evaluation metrics for loss changes for Gensim dataset with two relationships. 53
Table 5.12 Evaluation metrics for loss changes for FastText dataset with two relationships. 53
Table 5.13 Evaluation metrics for under-sampling for Gensim dataset with two relationships. 54
Table 5.14 Evaluation metrics for under-sampling for FastText dataset with two relationships. 55
Table 5.15 Evaluation metrics for over-sampling for Gensim dataset with two relationships. 57
Table 5.16 Evaluation metrics for over-sampling for FastText dataset with two relationships. 57
Table 5.17 Evaluation metrics for experiments for Gensim dataset with two relationships. 61
Table 5.18 Evaluation metrics for experiments for FastText dataset with two relationships. 61

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous Bag-of-Words
ELMo	Embeddings from Language Models
GPU	Graphics Processing Unit
NLP	Natural Language Processing
SMOTE	Synthetic Minority Over-sampling Technique

CONTENTS

1 INTRODUCTION....10
2 BACKGROUND....13
2.1 Machine learning....13
2.2 Word embeddings....15
2.3 Performance metrics....16
2.4 Data imbalance....18
2.4.1 Algorithm-level methods....19
2.4.2 Data-level methods....19
2.5 Knowledge models....22
2.6 WordNet....24
3 RELATED WORKS....27
4 METHODOLOGY....31
4.1 Development environment....31
4.2 Building the datasets....32
4.2.1 Methodology for creating the datasets....32
4.2.2 Datasets for classifying semantic relationships....35
4.3 Approach for classifying semantic relationships....38
4.4 Experiments....39
5 RESULTS....41
5.1 Experiments with three relationships....41
5.1.1 Baseline....41
5.1.2 Experiments with weighted loss and focal loss....42
5.1.3 Under-sampling experiments.....43
5.1.4 Over-sampling experiments....47
5.1.5 Comparing results....48
5.2 Experiments with two relationships....51
5.2.1 Baseline....51
5.2.2 Weighted loss and focal loss experiments....52
5.2.3 Under-sampling experiments.....54
5.2.4 Over-sampling experiments....57
5.2.5 Comparing results....57
6 CONCLUSION....62
REFERENCES....65

1 INTRODUCTION

Explicit knowledge models, like knowledge graphs, play a crucial role in storing, structuring, and interpreting information, making explicit the semantics underlying the domain knowledge and allowing for the processing of concepts, their properties, and the relations among them. However, building these models is a highly resource-demanding task, requiring the input of domain experts and knowledge engineers. In the last few years, the scientific community has proposed machine learning-based approaches to streamline this process by automating some tasks required for building explicit knowledge models. One of these tasks is classifying the semantic relationships between concepts (SUN et al., 2019; KHADIR; GUESSOUM; ALIANE, 2021; HOSSEINI et al., 2021), which has as input information regarding two related words or concepts, and classifies the relationship between these concepts. A possible approach for this task involves representing the related concepts with word embeddings, which are real-valued vectors that encode the semantics of words, and using these word embeddings as inputs for a machine learning approach that classifies the relation between pairs of concepts.

When considering word embeddings, it is important to differentiate between static and contextual embeddings. Contextual embeddings, such as those produced by language models like BERT (DEVLIN et al., 2019) and ELMo (PETERS et al., 2018), have the advantage of taking into account the context in which words are used. Thus, in this approach, a given word is represented by different embeddings in different contexts, which results in a more flexible representation of contextual variations of the semantics of a given word. However, the training process of these models tends to be heavier and more resource-demanding and produces computationally demanding models. In contrast, static embeddings, such as those developed by Mikolov et al. (2013) in the Word2Vec model, have the advantage of the models trained with them being relatively lighter, making them suitable for a wide range of applications and research.

In this scenario of classifying semantic relationships, it is common for datasets to be highly imbalanced. Data imbalance can harm the performance of machine learning models since they can be biased to prioritize majority classes, resulting in poor performance for minority classes. Although the literature provides several works that apply machine learning approaches to classify relationships from pre-trained static word embeddings that represent concepts, according to our knowledge, there are no works that systematically investigate the application of techniques to deal with data imbalance in this task.

In this context, our hypothesis is that by adopting strategies for mitigating the data imbalance we can improve the performance of machine learning approaches for this task. With this hypothesis in mind, we defined three goals for this work. The first one is defining a methodology for building datasets for classifying semantic relationships between concepts represented by pre-trained static word embeddings. The second one is building two datasets for this task. Finally, the third one is systematically applying some common approaches for dealing with data imbalance and evaluating the impacts of these techniques on the performance of machine learning models trained for this task.

In this work, we first defined a methodology for building datasets for the target task. The methodology takes as input a set of pre-trained static word embeddings and adopts the WordNet lexical resource as a reference for establishing semantic relationships between pairs of words in the word embeddings set. In order to do that, we defined a set of normalization rules and matching rules, to be used during the execution of the matching process for mapping the words in the word embeddings set and the words specified in WordNet.

After, we selected two different sets of word embeddings, with different characteristics and applied our methodology and a pipeline of pre-processing for building two different datasets with two different variations for the target task. Each variation of each dataset assumes a different set of relations. Finally, we selected a set of strategies for dealing with data imbalance and evaluated their impacts on the performance of a standard neural network trained for classifying semantic relationships.

In the experiments considering the datasets with three and two relationships, the majority of techniques failed to outperform the baseline in terms of F1 score. Although some methods, like SMOTE and its variants, showed potential in improving recall, particularly for the holonymy class, the baseline achieved the best performance, in general. Furthermore, findings from experiments using two relationships mimicked those from the three relationships dataset, but with marginally better F1 scores. Differences between recall and precision were evident, showing the challenge of optimizing one without degrading the other. Future studies might explore leveraging contextual word embeddings for this task and devising strategies to address dataset imbalance specifically designed for this task.

The remainder of the work is organized as follows. In Chapter 2, we detail the necessary background to understand this work, providing an overview of machine learning, word embeddings, performance metrics, data imbalance techniques, knowledge models and WordNet. In Chapter 3, we discuss some studies that are related to the task of classifying semantic relations using word embeddings. In Chapter 4, we detail the work's methodology, describing the development environment, the datasets used and the processing we performed on them, the neural network architecture used in our experiments, the experiments performed, and the metrics chosen to evaluate the experiments. In Chapter 5, we analyze the results of our experiments. And, in Chapter 6, we summarize the conclusions of this work and present the possibilities for future work.

2 BACKGROUND

In this chapter, we present the necessary background to understand this work. Section 2.1 presents an overview of machine learning and neural networks. Section 2.2 presents an overview of word embeddings and their use in natural language processing tasks. Section 2.3 presents a set of performance metrics that can be used to evaluate machine learning models in classification tasks. Section 2.4 details the set of approaches designed for dealing with data imbalance that were evaluated in this work. Section 2.5 discusses knowledge models, such as ontologies and knowledge graphs. Finally, Section 2.6 presents WordNet, a lexical database of words.

2.1 Machine learning

According to Russell and Norvig (2020), machine learning is a subset of artificial intelligence (AI) that involves the use of statistical techniques to enable machines to improve their performance on a task with experience and feedback. There are three main types of feedback that define the three main types of learning for machine learning models: supervised learning, unsupervised learning and reinforcement learning.

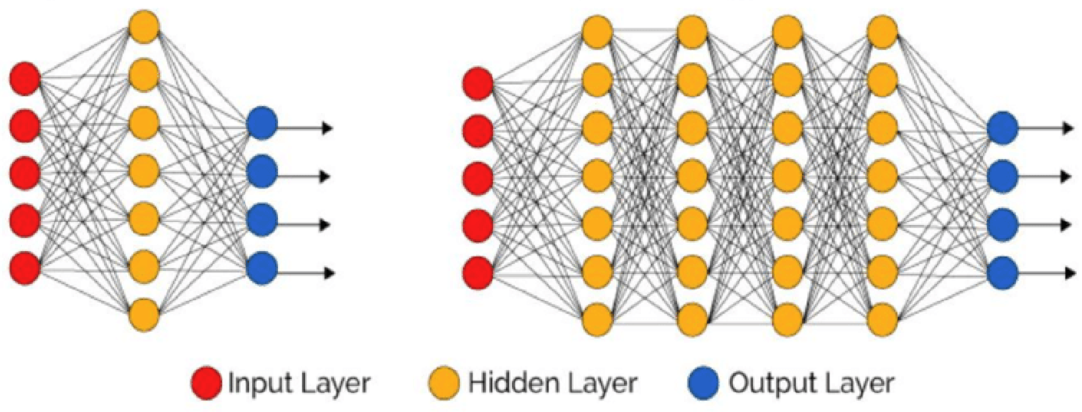
Supervised learning is a type of learning where a model is created to make predictions based on a set of labeled training data. This means that each example in the training data comes with a corresponding output value or label. The goal in this scenario is to use this labeled data as input for training a model that is able to map unseen samples to their corresponding labels. A common example of supervised learning is classifying images according to the class of objects in the image, such as classifying images of handwritten digits.

Unsupervised learning, on the other hand, involves training a model using data that is not labeled. The model is tasked with finding patterns within the data. A common application of unsupervised learning is clustering (grouping similar data together), with an example being grouping different customers of a store into different profiles, based on their preference for certain types of products.

In reinforcement learning, an agent learns to make decisions by taking actions in an environment to achieve a goal. The agent learns by trial and error, receiving rewards or penalties for its actions. Over time, the agent learns a policy, which is a strategy for choosing actions that maximize the total reward. An example of an application of reinforcement learning is training an agent to play games, like Go and chess.

This work focuses on a supervised learning context and adopts neural networks for classifying semantic relationships. Neural networks, often referred to as artificial neural networks, are inspired by neural networks of the brain and in the learning process of those systems to make predictions and decisions. A neural network is a collection of nodes connected by weighted links. In general, nodes are organized in layers, in a way that nodes in a given layer are connected to nodes of adjacent layers. Neural network architectures are organized in such a way that the input layers receive input data, the output layer represents the predictions associated with the input, and the intermediate layers carry out the processing. In this process, in traditional neural network architectures, the input data is multiplied by the weights between the units, and this process repeats across the layers. The learning process in a neural network involves adjusting the weights and biases of the network to minimize the error between the network's output and the desired output (GOODFELLOW; BENGIO; COURVILLE, 2016). Backpropagation is the algorithm enabling this adjustment. It works by computing the gradient of the cost function with respect to each weight, allowing for efficient weight updates in the direction that reduces the error. This backward flow of adjustments, after the process of forward propagation, ensures an iterative refinement of the model's predictions. In Figure 2.1 we can see two examples of neural networks with an input layer, an output layer, and hidden layers. The neural network on the left has just one hidden layer and the one on the right has four hidden layers.

Figure 2.1: Representation of neural networks.



Source: Dastres and Soori (2021)

One mechanism that is used to quantify how far the predicted output is from the actual output is the loss function, also known as the cost function. The purpose of a loss function is to guide the process of learning the weights in a neural network. One function that is commonly used for classification problems is the cross-entropy loss function (GOODFELLOW; BENGIO; COURVILLE, 2016). The cross-entropy loss function is described by Lin et al. (2017) as $CE(p_t) = -log(p_t)$, in which p_t is the model's estimated probability for the class with label t.

2.2 Word embeddings

Word embeddings are n-dimensional vectors of real words that represent the semantics of words, in a way that different dimensions capture different aspects of the word's meaning. They are used to capture semantic and syntactic relationships between words, based on the context in which they appear. They can be used in natural language processing tasks to represent text for machine learning models.

It is important for this work to differentiate between static and contextual embeddings. The main difference between them is that contextual embeddings have the advantage of taking into account the context in which words are used, meaning that the same word is represented by different word embeddings in different contexts. Contextual embeddings such as those produced by BERT (DEVLIN et al., 2019) and ELMo (PETERS et al., 2018) are able to capture many syntactic and semantic properties of words under diverse linguistic contexts by considering the context of a sentence during training (LIU; KUSNER; BLUNSOM, 2020). ELMo, for example, uses a bidirectional language model that takes into account at each of its layers, contextualized representations which are the concatenation of the left-to-right and right-to-left representations, obtaining N representations, for a sequence of length N (LIU; KUSNER; BLUNSOM, 2020).

Static embeddings obtain a single global representation for each word, ignoring their context (LIU; KUSNER; BLUNSOM, 2020). Word2Vec is a popular model for creating static word embeddings. It was developed by researchers at Google, and provides two architectures to generate word embedding: Continuous Bag-of-Words (CBOW) and Skip-Gram (MIKOLOV et al., 2013). The difference between those models is that the CBOW architecture predicts the current word based on the context (surrounding words), and the Skip-gram architecture predicts surrounding words given the current word, with both models trying to maximize the classification of a word based on another word in the same sentence during training.

FastText[1] and Gensim[2] are both popular open-source libraries for natural language processing and machine learning that are used to create static word embeddings. FastText, based on Skip-gram takes into account subword information by representing each word as a bag of character n-grams. An n-gram is a contiguous sequence of n items from a given text, for example, taking the word *where* and $n = 3$ as an example, it will be represented by the character n-grams: <*wh, whe, her, ere, re*> (BOJANOWSKI et al., 2017a). A vector representation is associated with each character n-gram, with words being represented as the sum of these representations. Gensim, on the other hand, is a Python library that offers tools for topic modeling, document indexing, and similarity analysis, and can be used to generate standard Word2Vec embeddings, without considering subword information (but it also supports the creation of models that consider such information).

Word embeddings are commonly used in several natural language processing problems. They are used to feed word-level input to neural networks for tasks such as text classification, sentiment analysis, entity recognition, etc (JURAFSKY; MARTIN, 2023).

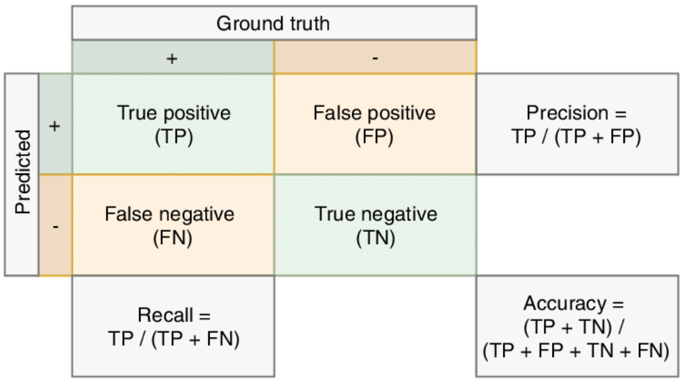
2.3 Performance metrics

In machine learning, performance metrics are used for evaluating the effectiveness of a machine learning approach in a given task. The selection of performance metrics depends largely on the problem at hand. The following metrics have their own advantages and shortcomings, especially regarding data imbalance, as they provide different perspectives on the performance of a classifier. To understand the metrics shown in this section it's necessary to understand the concept of a confusion matrix.

A confusion matrix is a table that is often used to describe the performance of a classification model. Figure 2.2 represents a confusion matrix, where each column represents the instances of an actual class and each row represents the instances of a predicted class. The main diagonal represents correct predictions (true positive and true negative results), while elements outside of the main diagonal represent incorrect predictions (false positive and false negative results). The main metrics we can calculate from a confusion matrix are accuracy, precision, recall and F1 score. Traditionally, confusion matrices and the metrics precision, recall, and F1 score are defined for binary classification problems, being presented in the following paragraphs as described by Faceli et al. (2011).

[1]<https://fasttext.cc/>
[2]<https://pypi.org/project/gensim/>

Figure 2.2: Confusion matrix with evaluation metrics.



Source: Jeppesen et al. (2019)

Accuracy is the simplest evaluation metric. It's the ratio of the number of correct predictions over the total number of input samples. It works well if there are an equal number of samples belonging to each class. But if we have an imbalanced dataset this metric can be deceiving, as it is possible that a model has great accuracy but misses most samples of the minority class. It can be calculated from the confusion matrix with the following formula:

$$accuracy = (TP + TN)/(TP + FP + TN + FN) \tag{2.1}$$

Precision is the fraction of the total amount of correct classifications for the positive class over the number of instances that were predicted as being from the positive class. It's a measure of a classifier's exactness. It can be calculated from the confusion matrix with the following formula:

$$precision = TP/(TP + FP) \tag{2.2}$$

Recall is the fraction of the total amount of correct classifications of the positive class among the number of instances of that class. It's a measure of a classifier's completeness. It can be calculated from the confusion matrix with the following formula:

$$recall = TP/(TP + FN) \tag{2.3}$$

A summary of both Precision and Recall is the F1 score; a single number that is defined as the harmonic mean of precision and recall. It is especially useful in the case of an uneven class distribution because it balances the two previous measures. It can be calculated with the following formula:

$$F1 = 2 * P * R/(P + R) \tag{2.4}$$

In the context of multi-class classification, the common approach is to calculate the metrics for each class and then average the values, considering the class as the positive class in a binary problem, and considering all other classes as negative. Some options to average the precision, recall, and F1 score metrics are macro-averaging, micro-averaging, and weighted-averaging. Macro-averaging will compute the metric independently for each class and then take the average treating all classes equally, micro-averaging will aggregate the contributions of all classes to compute the average metric, and weighted-averaging will compute the metric independently for each class and then take the average considering the number of instances for each classes.

2.4 Data imbalance

Handling imbalanced datasets is a common challenge in machine learning. Common scenarios with this problem are credit card fraud and spam e-mail detection. In these cases, the positive class instances - fraud transactions or spam e-mails - are far fewer than negative class instances - non-fraud transactions and non-spam e-mails. If we apply a machine learning approach to detect the positive class instances, the models might perform poorly, since the imbalanced data has a bias towards the negative class, which has many more instances, leading to a high number of false negatives.

To address these issues, the literature provides several approaches, which can be classified into *algorithm-level* methods and *data-level* methods, according to Leevy et al. (2018). Algorithm-level methods try to mitigate data imbalance by algorithmically changing the learning process. These approaches can be divided into ensemble methods and cost-sensitive methods. Ensemble methods, such as bagging and boosting, combine multiple classifiers to determine the output of the classification task. Cost-sensitive methods try to assign more weight to an instance that is misclassified. Data-level methods, on the other hand, try to mitigate data imbalance by changing the data available for training the models. These methods can be divided into feature selection methods and data-sampling methods, such as over-sampling and under-sampling.

The following sections present some approaches considered in this work. Firstly presenting the algorithm-level approaches and, after, presenting the data-level approaches.

2.4.1 Algorithm-level methods

Two common algorithm-level methods we can use to improve the learning process for neural networks with imbalanced datasets are using a *weighted loss function* and a *focal loss function*.

A weighted loss function uses different weights that are assigned to different classes in the loss function. The weight is usually inversely proportional to the class frequency, meaning that higher weights are assigned to the minority class and lower weights are assigned to the majority class, in a binary classification problem. This means the model is penalized in a higher degree for incorrectly predicting the minority class (ELKAN, 2001). The formula for a weighted cross-entropy loss function as described by Lin et al. (2017) is $CE(p_t) = -\alpha_t log(p_t)$. This formula adds α_t, a weighting factor, to the cross-entropy loss, where p_t is the model's estimated probability for the class with label t.

The idea of the focal loss function technique, on the other hand, is to assign more importance to hard-to-classify instances and less importance to easy instances. This is achieved by adding a factor to the cross-entropy function which decreases the loss contribution from easy-to-classify examples (LIN et al., 2017). Its formula, as described by Lin et al. (2017) is $FL(p_t) = -\alpha_t(1 - p_t)^\gamma log(p_t)$. This formula adds a modulating factor to the weighted cross-entropy loss with a parameter $\gamma >= 0$, where p_t is the model's estimated probability for the class with label t, and α_t is the weighting factor for the class.

2.4.2 Data-level methods

For data-level methods, common approaches involve under-sampling the majority class (or classes if the problem classifies more than two classes) so we can have a more balanced dataset, or over-sampling the minority class (or classes), creating new instances for those minority classes. In this work, we chose to focus on random-under-sampling and Tomek links under-sampling as under-sampling techniques, and on three techniques that are a variation of SMOTE, for over-sampling.

Random under-sampling is a technique where instances of the majority class are randomly eliminated until a desired balance between the majority and minority class is achieved (HE; GARCIA, 2009). One common pitfall of this technique is that, since we are removing instances randomly, it is possible that the removed instances may cause the classifier to miss important concepts pertaining to the majority class (HE; GARCIA, 2009).

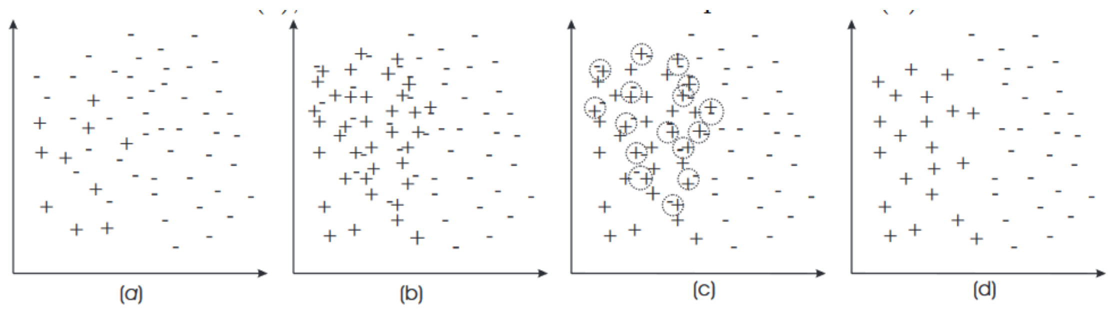
Tomek links are pairs of instances that are the closest to one another but are of opposite classes (TOMEK, 1976). Different from random under-sampling, this technique tries to preserve the classification frontier, but it requires many additional computations when compared to random under-sampling, which can have a huge impact on big datasets, taking 10 times more to execute for our largest datasets.

Synthetic Minority Over-sampling Technique (SMOTE) is an over-sampling method that works by creating synthetic samples from the minority class by selecting nearest neighbor instances from the same class for a given instance and creating new instances between the given instance and those by a random amount within the difference between them (HE; GARCIA, 2009). One of the advantages of SMOTE is that it makes the decision regions larger and less specific (HAN; WANG; MAO, 2005).

The SMOTE-Tomek technique is a combination of over-sampling of the minority class using SMOTE and then under-sampling the class using Tomek links (BATISTA; BAZZAN; MONARD, 2003). The algorithm tries to perform a cleanup of the new over-sampled instances by removing the Tomek links. One problem with combining these two algorithms is that this technique takes a long time to run, making it by far the most costly technique we used. The algorithm process can be seen in Figure 2.3, with (a) showing the original dataset, (b) showing an over-sampled dataset using SMOTE, (c) showing the Tomek links found, and (d) showing the dataset after removing the Tomek links.

The Borderline SMOTE Technique is a variant of SMOTE, which takes advantage of the fact that examples far from the borderline may contribute little to a classification task. Instead of over-sampling the minority class and taking into consideration the nearest neighbors of each example, the algorithm generates synthetic examples in the borderline space where the majority class examples may intrude into (HAN; WANG; MAO, 2005). In order to do this, the algorithm determines the m-nearest neighbors for each instance of the minority class, and then the instances are classified into three categories: as noise, if all its neighbors are from another class; as safe, if less than half of its neighbors are from its own class; or as borderline (or in danger), if half or more than half of its neigh-

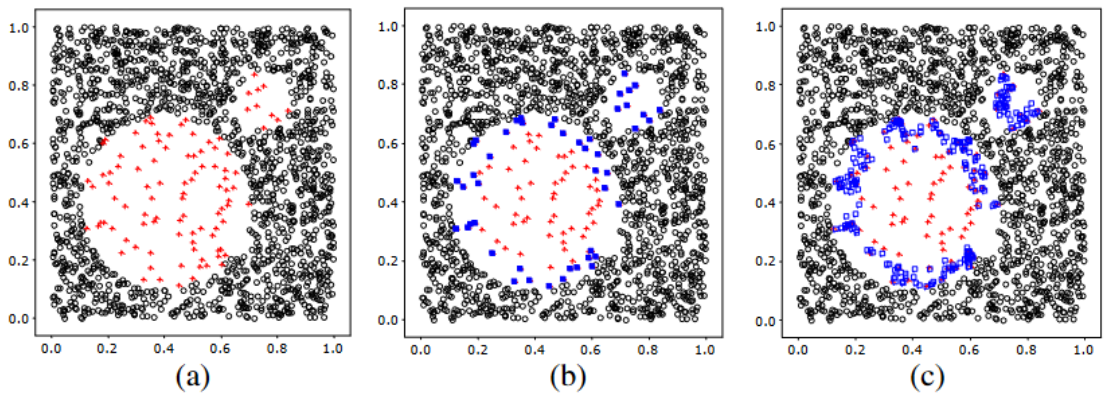
Figure 2.3: SMOTE-Tomek: (a) original data set ; (b) over-sampled data set; (c) Tomek links identification; and (d) borderline and noise examples removed.



Source: Batista, Bazzan and Monard (2003)

bors are from another class. For those borderline instances, new instances are created by selecting the k-nearest neighbor instances from the minority class and creating new instances between the given instance and those by a random amount within the difference between them. The algorithm process can be seen in Figure 2.4, with (a) showing the original dataset, (b) showing the borderline examples in the minority class, and (c) showing the dataset after over-sampling the minority class with the Borderline SMOTE algorithm, with new instances being displayed with hollow squares.

Figure 2.4: Borderline SMOTE: (a) The original distribution of an example dataset. (b) The borderline minority examples (solid squares). (c) The borderline synthetic minority examples (hollow squares).



Source: Han, Wang and Mao (2005)

2.5 Knowledge models

Explicit knowledge models play a crucial role in storing, structuring, and interpreting information, making explicit the semantics underlying the domain knowledge and allowing for the processing of concepts, their properties, and the relations among them. Common types of knowledge models are ontologies and knowledge graphs.

An ontology is a formal, explicit specification of a shared conceptualization (BORST, 1999). In simpler terms, it is a formal definition, with the use of a formal language, which guarantees that the ontology is unambiguous and machine-readable, with a vocabulary that is mapped to elements of a conceptualization, a simplified, generalized view of an observed universe or part of the universe (KONOPKA, 2015). An ontology, then, has three main elements: a set of concepts, formal axioms that constrain the semantics of those concepts, and definitions that define concept meanings (lexicon). It is important to note that ontologies focus on universal concepts of the domain, not describing specific instances from it.

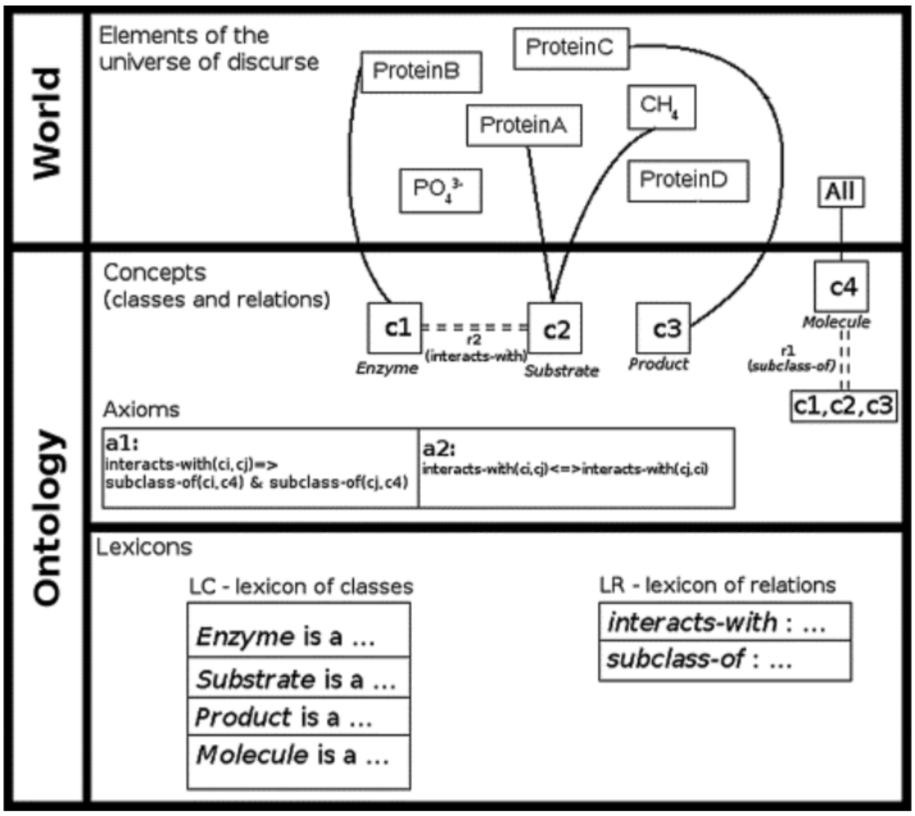
An example of an ontology of a set of molecules can be seen in Figure 2.5. In the example, the ontology comprises 6 concepts: with four classes of elements (*Enzyme*, *Substrate*, *Product*, *Molecule*) and two relations (*interacts-with* and *subclass-of*). There are two axioms included in the ontology Axiom *a1* says that only objects that are molecules can interact. Axiom *a2* denotes that the *interacts-with* relation is symmetric. the ontology also comprises two lexicons: *LC* lists the definitions of classes, *LR* lists the definitions of relations. The description of the universe is done through annotation. In the figure, elements of the universe (*ProteinA*, *ProteinB*, *ProteinC*, *ProteinD*, CH_4, ${PO_4}^{3-}$) were annotated with terms from the ontology, which is marked with thick black lines between elements and concepts. It is also worth noting that there's a class hierarchy in the ontology, with *Molecule* being the most general term, and *Enzyme*, *Substrate*, *Product* being the specific cases of a molecule.

A knowledge graph is a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities (HOGAN et al., 2021). Common examples of knowledge graphs are DBpedia[3], aiming to structure data from Wikipedia, and Wikidata[4], a source of open data that Wikipedia uses. Knowledge graphs can be un-

[3]<https://www.dbpedia.org/>

[4]<https://www.wikidata.org/wiki/Wikidata:Main_Page>

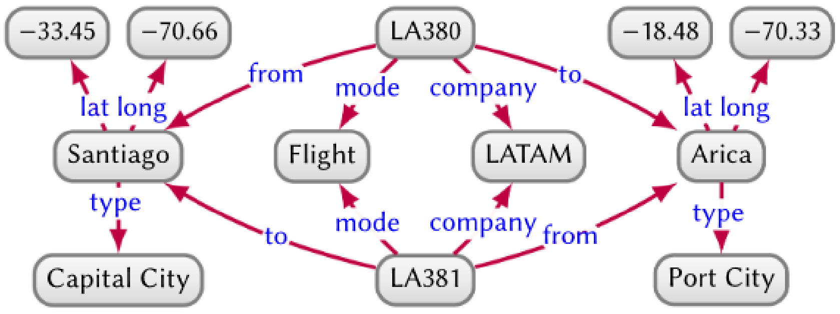
Figure 2.5: Example of an ontology.



Source: Konopka (2015)

derstood as a graph that represents knowledge, which can include an ontology as well as information regarding instances, which is structured according to the schema provided by the ontology. They gained popularity when Google announced their knowledge graph to enhance search engine results (JI et al., 2022). Figure 2.6 shows an example of a knowledge graph for flight data.

Figure 2.6: Example of a knowledge graph for flight data.



Source: Hogan et al. (2021)

2.6 WordNet

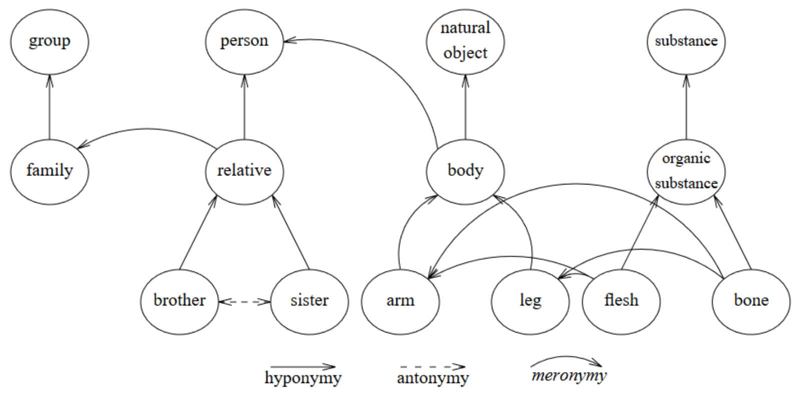
WordNet (MILLER, 1995) is a large lexical database of words in English, developed by the Cognitive Science Laboratory at Princeton University. It groups words into sets of synonyms denoting a lexical concept, called *synsets*, provides their definitions, and records the various semantic relations between these synonym sets. The purpose of WordNet is to facilitate natural language processing (NLP) tasks.

The structure of WordNet distinguishes between different types of relations for different lexical categories. For nouns, the main relations include hypernymy (super-name), hyponymy (sub-name), meronymy (part-name), and holonymy (whole-name). For verbs, the key relations are hypernymy (super-name), troponymy (manner-name), and entailment relations. Adjectives and adverbs are linked via synonymy and antonymy relations. The definitions of the relationships mentioned throughout this work provided by Miller et al. (1990) and Khadir, Guessoum and Aliane (2021) are as follows:

- Synonymy: Two expressions are synonymous in a linguistic context if the substitution of one for the other in that context does not alter the truth value in the context.

- Antonymy: Lexical relationship between word forms, not a semantic relation between word meanings, denoting that one word represents the opposite of another word.
- Hypernymy/Hyponymy: Semantic relation between word meanings. A concept represented by the synset $\{x, x', ...\}$ is said to be a hyponym of the concept represented by the synset $\{y, y', ...\}$ if native English speakers accept sentences constructed from such frames as "An x is a (kind of) y". In such example we define the synset $\{y, y', ...\}$ as the hypernym of synset $\{x, x', ...\}$. A specific kind of hypernymy relationship is the instance hypernymy relationship, denoting that an expression is an instance of another, such as "Barrack Obama" is an instance of "President".
- Meronymy/Holonymy: A concept represented by the synset $\{x, x', ...\}$ is a meronym of a concept represented by the synset $\{y, y', ...\}$ if native English speakers accept sentences constructed from such frames as "A y has an x" (as a part) or "An x is a part of y" . In such example we define the synset $\{y, y', ...\}$ as the holonym of synset $\{x, x', ...\}$. The holonymy relationship can be split into further relationships, such as part holonymy, denoting that a synset is a component of another, such as a "branch" is a component of a "tree", member holonymy, denoting that a synset is an element of another, such as a "tree" is an element of a "forest", and substance holonymy, denoting that a synset is a material of another, such as "wood" is a material of a "door".
- Similarity: The relationship exists between primary adjectives and their corresponding satellite adjectives. Satellite adjectives are those that share a semantic link with central adjectives. For instance, "accurate" is a primary adjective that is linked with the satellite adjectives: "straight", "surgical", "true", and "veracious".

Figure 2.7: Network representation of hyponymy, antonymy and meronymy relations among an illustrative variety of lexical concepts.



Source: Miller et al. (1990)

3 RELATED WORKS

In this chapter, we give a brief overview of works in the literature that deal with classifying semantic and lexical relations using word embeddings. Most of the work we could find used relations that were domain-specific instead of focusing on domain-independent relations, such as hypernymy and holonymy. Also, of the few that we could find that used the relations present in this work, many were focused on relation classification tasks for a specific domain.

We used the work from Khadir, Guessoum and Aliane (2021) as the main reference for our work. In this work, the authors present a neural network model to classify eight types of relations: hypernymy, instance hypernymy, part holonymy, member holonymy, substance holonymy, similarity, antonymy, synonymy. They use WordNet to extract relations from all synsets in the dataset and then replace the involved words with their pre-trained word vectors with 300 dimensions using pre-trained Skip-gram word embeddings. For the matching of words to word embeddings, they also perform some format changes such as adding/deleting spaces or hyphens (-) between words that compose the same lemma to match the way they may be written in the word embeddings set. In the end, they had a dataset with 307,856 pairs. They adopt a neural network model with two hidden layers that contain 512 neurons each and use the ReLU activation function. After each hidden layer, a dropout of 0.2 is applied to prevent overfitting. They use the RMSprop optimizer with a batch size of 128 and categorical cross-entropy as the loss function. The output layer uses the softmax activation function. They made three experiments, one with the entire dataset which resulted in 0.84 F1 score, one with only the hypernymy, part holonymy, antonymy and synonymy classes, which resulted in 0.78 F1 score and a third experiment for binary classification with a hypernymy class and a non-hypernymy class, which resulted in 0.87 accuracy. They note that some classes had much better performance than others. According to them, a possible explanation is due to possible semantic overlapping in some cases, as they ignored ambiguity in the experiments. Our work tries to be an improvement on top of this paper by proposing an improved methodology for building datasets that removes ambiguity in the dataset, doing tests with another pre-trained word embeddings set, and using more matching rules evaluating data imbalance techniques to improve performance.

Another related work is by Sun et al. (2019). In their work, they explore machine learning techniques to evaluate suggested changes in an existing biomedical ontology.

They use a hybrid convolutional neural network and multi-layer perceptron classifier using a combination of graphs, concept features, and word embeddings to classify hypernymy relations. They also use Skip-gram Word2Vec word embeddings, but with word embeddings with 200 dimensions. To perform the matching of words to word embeddings they perform many format changes such as removing punctuation, removing digits, transforming words to lowercase, and performing stemming. They find that their model performs really well, achieving an impressive 0.972 precision score for the task of predicting a hypernymy relationship (IS-A relation). It is important to notice that this work focuses on classifying relations for the biomedical domain, which might explain the much higher performance results compared to other papers. The high performance can also be explained by the fact that this work considers a richer set of information as input, instead of using only word embeddings. Some of their ideas for transforming words for word embedding matching were also applied to our work.

In Hosseini et al. (2021), the authors propose an automated approach to infer semantic relations among concepts and construct an ontology to help requirements authors in the selection of the most appropriate information type terms for privacy policies. For this, they utilize word embeddings and convolutional neural networks to classify information type pairs as either hypernymy, synonymy, or unknown. They gathered data from the Google Play Store and trained their own Skip-gram Word2Vec word embeddings. They perform tests using a weighted loss function to account for imbalance in ontological relations, and consider using over-sampling as a next step for their work. For the hypernymy instances they do not only consider direct relations, but also gather the indirect hypernyms for creating their dataset. Their model results in a 0.904 F1 score. Even though this work didn't consider a comprehensive set of methods to combat data imbalance, they used a weighted loss function and suggested trying over-sampling, which we both tested in our work.

Similarly to the approach proposed in Hosseini et al. (2021), another work that creates a label to represent the lack of a relation between words is Chen et al. (2020), and in that the authors find that having a label to represent a lack of relationship results in diminished relation extraction performance. We didn't consider a label to represent the absence of relations between concepts in our work because even though a pair of words isn't classified as having a certain relation we can't say they don't have one of the selected relations, because the dataset can be incomplete.

In Gasmi, Laval and Bouras (2019), the authors predict instances of classes in a cybersecurity ontology and the semantic relationship between them using a Long Short Term Memory model and Word2Vec embeddings. In their work, they obtain data from the National Vulnerability Database, and also use conversion rules such as converting words to lowercase before word embedding matching, and converting digits to 0. The model shows an F1 score of 0.79. Even though in this work they consider different relations from ours, we included it because they found that their model seems to suffer from overfitting, perhaps needing more data, which is something that we also found was true from our tests with our models initially, and seems to occur in other papers as well. Besides that, this work also proposes other methods of matching words to their own word embedding set.

In Oussaid, Bouarab-Dahmani and Cullot (2022), the authors propose an approach that aims to automate the extraction of new ontological concepts from unstructured data with the goal of enriching a food ontology. For this purpose, an ontology and a corpus of food data have been built. The data is obtained from web scraping from Wikipedia (Wikipedia contributors, 2004) and is used to train a Skip-gram Word2Vec model. They also pre-process their data with lowercase conversion, removal of special characters and lemmatization (representing each word by its lemme). The obtained results show a precision score of 0.78. The interesting aspect of this work is the fact that they trained their own word embeddings to create a food ontology. We didn't try creating our own word embeddings, given that it is an extremely costly process that requires resources that are not available in our project.

In Lezama-Sánchez, Vidal and Reyes-Ortiz (2022), the authors present three embedding models based on semantic relations extracted from Wikipedia (Wikipedia contributors, 2004) to classify texts using sentence patterns between related words. For testing the models they use a convolution neural network. The relations they choose to classify are synonymy, hyponymy and hypernymy. One interesting technique they used to augment their dataset was to create more synonyms, by creating another pair inverting the order of the words in the relationship pair, which is a technique that we also applied in our dataset. They compare their word embedding models with other existing models (GloVe, FastText, and WordNet-based) and find that it performs better in the classification task.

As we can see there are many works that try to classify semantic relations using word embeddings, but many of these try to perform that task in a specific domain, such as biomedicine. Furthermore, we couldn't find works that systematically evaluated data imbalance techniques to mitigate the high level of imbalance that is common in the task of classifying general semantic relationships. We also noticed that many of the papers developed different matching techniques to create the datasets for training their machine learning models. With that in mind, we propose another method to perform such word matching and perform experiments for evaluating different data imbalance techniques to mitigate such issues.

4 METHODOLOGY

In this chapter, we present the methodology of this work. Section 4.1 details the development environment for the experiments. Section 4.2 presents the proposed methodology for building datasets for semantic relationship classification and the datasets that we developed following the proposed methodology. Section 4.3 details our approach for classifying semantic relationships, describing the neural network architecture that we adopted in our experiments. Section 4.4 details the different experiments we performed.

4.1 Development environment

The development of this work was done using the programming language Python[1]. Most of the scripts developed were run locally, with some training and validation of the neural network models performed using the Google Colaboratory environment[2] with its free tier.

We also used many Python libraries to aid the development done in this work. We mainly used the libraries Tensorflow[3] and Keras[4] for training and defining the neural network models, the library nltk[5] to access the WordNet database and the library imbalanced-learn[6] for its techniques to mitigate data imbalance in our datasets. Other supporting libraries such as scikit-learn[7], numpy[8], pandas[9], seaborn[10], focal-loss[11], matplotlib[12] and textblob[13] were used as well.

[1]<https://www.python.org/>
[2]<https://colab.research.google.com/>
[3]<https://www.tensorflow.org>
[4]<https://keras.io/>
[5]<https://www.nltk.org/>
[6]<https://imbalanced-learn.org/stable/>
[7]<https://scikit-learn.org/stable/>
[8]<https://numpy.org/>
[9]<https://pandas.pydata.org/>
[10]<https://seaborn.pydata.org/>
[11]<https://focal-loss.readthedocs.io/en/latest/>
[12]<https://matplotlib.org/>
[13]<https://textblob.readthedocs.io/en/dev/>

4.2 Building the datasets

In this section, we present the proposed methodology for creating the datasets and, besides that, we present the datasets that we developed following our methodology for carrying out our experiments. Section 4.2.1 describes our methodology proposed for building datasets for our target task. Section 4.2.2, on the other hand, presents the datasets that we developed following our methodology.

4.2.1 Methodology for creating the datasets

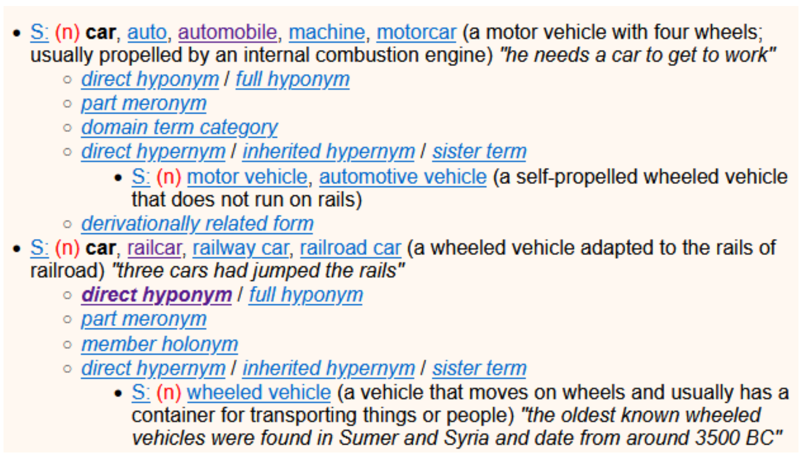
As previously mentioned, in this work we propose a methodology for building datasets for our target task. It is important to notice that the proposed methodology is agnostic regarding the technique used to generate the word embeddings from the words.

The proposed methodology takes as input a set of pre-trained word embeddings, which consists of a list of tuples (W, E). In these tuples, W is a word and E is its corresponding pre-trained static word embedding (an n-dimensional vector of real numbers). In other words, it's a list of words with their respective corresponding word embeddings. With that in mind, the creation of the datasets can be divided into four parts: obtaining the words that will be used to search for related words, searching for pairs of related words, processing these pairs, and finally creating the dataset using word embeddings.

For the list of words, all the words from each Synset (set of synonyms) from WordNet are downloaded using the nltk library, and then, for each word on the list, we search for related words using a set of chosen relationships (any set of relationships from WordNet can be chosen). For the search of related words, given a word from the list, we search for related words considering all possible meanings for the search word in WordNet. That is, when a related synset was found, all words that could represent that related synset were included as related words. For example the word "car" has multiple meanings, such as "a motor vehicle with four wheels; usually propelled by an internal combustion engine" or "a wheeled vehicle adapted to the rails of railroad", for the first meaning a direct hypernym would be "motor vehicle", and for the second a direct hypernym would be "wheeled vehicle", both of those were included as hypernyms of the word "car". This process of agglutinating the multiple meanings of words is done specifically for static word embeddings, as we have only one embedding for each word, even though it may affect performance, as we are losing the different meanings of words. In Figure

4.1 we can see those definitions and those direct hypernyms.

Figure 4.1: Example of different meanings of the word "car".



- S: (n) **car**, auto, automobile, machine, motorcar (a motor vehicle with four wheels; usually propelled by an internal combustion engine) *"he needs a car to get to work"*
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *domain term category*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) motor vehicle, automotive vehicle (a self-propelled wheeled vehicle that does not run on rails)
  - *derivationally related form*
- S: (n) **car**, railcar, railway car, railroad car (a wheeled vehicle adapted to the rails of railroad) *"three cars had jumped the rails"*
  - ***direct hyponym*** / *full hyponym*
  - *part meronym*
  - *member holonym*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) wheeled vehicle (a vehicle that moves on wheels and usually has a container for transporting things or people) *"the oldest known wheeled vehicles were found in Sumer and Syria and date from around 3500 BC"*

Source: WordNet Search[14]

After finding the related words for each of the words from WordNet, a process of completion of the pairs for symmetric relations is done for the chosen relationships. For example, for the pairs of words related by the synonymy relationship, if that relationship is included. Thus, if we have a pair of related words A and B, we included the pair B and A, if not already included.

After obtaining those related words, the dataset generated will possibly contain pairs of words that are related by more than one relationship, resulting in a multi-label dataset. After that, each word from the pairs from the previous phase is transformed into its equivalent word embedding using a set of word embeddings, and the relationships between the pairs are replaced by numbers that represent them.

To search for words in the set of word embeddings, we use *normalization rules* that are applied to all words before the matching process begins, and *matching rules* that create variations of the word during the execution of the matching process in case the word is not normally found, in order to increase the probability of matching. When a word from the pair of related words was not in the set, then the pair is not included in the dataset. It is also important to note that we singularized the words in the lists of word embeddings, if the singular version of the word was not already present in the data.

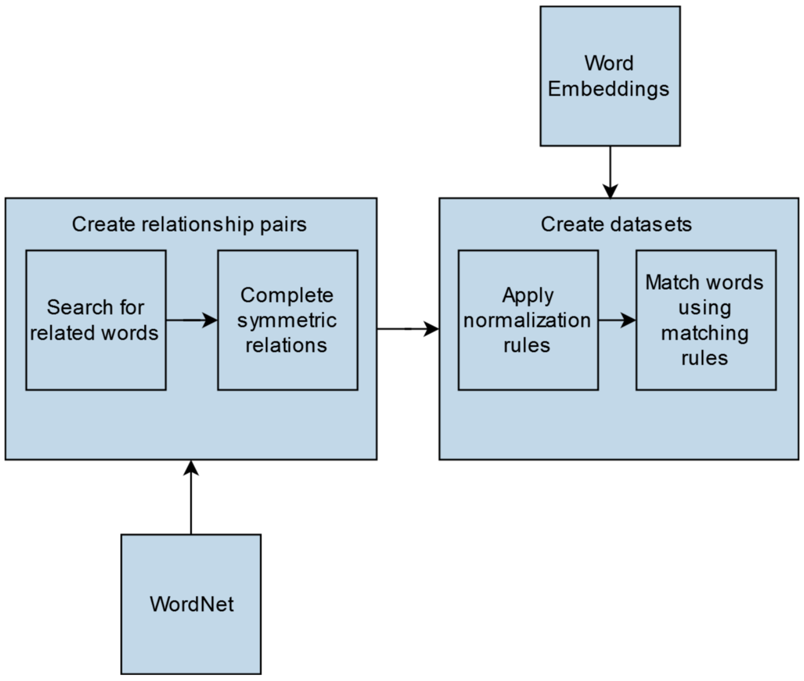
- Normalization Rules:
  - Normalize everything to lowercase;
  - Replace all special characters, characters that are not letters or numbers, with

’_’ (the ’_’ was chosen as the replacement for special characters because it was commonly used for WordNet in nltk).

- Matching rules:
  - Replace last letter, if it is a ’y’, with an ’e’ (ex: antecedency and antecedence);
  - Remove last letter, if it is a ’y’, and add ’ies’ (plural);
  - Add an ’s’ at the end (plural);
  - Replace ’s’ with ’z’ and ’z’ with ’s’ (differences between American and British English);
  - Add/remove a ’d’ at the end (ex: telecommunicate and telecommunicated).

The rules were created experimentally by analyzing data from words that had the smallest normalized distance value but weren’t a match. For this we used the hamming distance (HAMMING, 1950) of the words divided by the length of the longest word in the comparison. During this matching process, it was checked if a pair had already been inserted in the dataset, to avoid duplicates. The complete transformation pipeline is described in Figure 4.2.

Figure 4.2: Processing pipeline.


Word Embeddings
Create relationship pairs
Search for related words
Complete symmetric relations
Create datasets
Apply normalization rules
Match words using matching rules
WordNet

Source: The Author

4.2.2 Datasets for classifying semantic relationships

Notice that the methodology defined in Section 4.2.1 considers two parameters for the dataset creation: the set of relationships selected and the set of word embeddings used. In this work, we selected the following relationships:

- Hypernymy;
- Holonymy;
- Synonymy.

It is important to note that in Khadir, Guessoum and Aliane (2021), the study that we are using as a reference, has datasets that use those same relations and include also the following relations: similarity, antonymy, instance hypernymy relationships. Besides that, the authors also split the holonymy relationship into three specific subtypes: substance holonymy, part holonymy and member holonymy relationships. We joined these three holonymy relationships in a single relationship category because we are focusing on general relations between parts and wholes, and because this category is the least represented in our datasets. We also decided to remove from our dataset the instance hypernymy relationship, because we are focusing on conceptual relationships and not on instance relationships. Besides that, we removed the similar and antonymy relations, because these relationships are not as relevant to the construction of structured knowledge bases as hypernymy, holonymy, and synonymy.

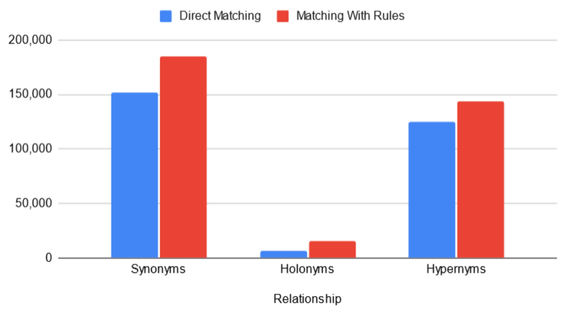
For creating our datasets we used two sets of pre-trained word embeddings. The first pre-trained word embeddings dataset[15], was trained using Gensim[16] with the Skip-gram algorithm, with the English Wikipedia Dump of February 2017 corpus[17] and was made available by the Language Technology Group of the University of Oslo[18]. It has a vocabulary of 228,670 words, which were obtained using a lemmatization process and produces word embedding vectors with 300 dimensions. It is worth noting that the process of lemmatization for the words in this dataset may affect performance in the classification task, similar to the process of agglutinating word meanings described in Section 4.2.1, as we are losing the different meanings of words. The second dataset [19], is the same dataset used in our reference paper (KHADIR; GUESSOUM; ALIANE, 2021). The vectors were

[15]<http://vectors.nlpl.eu/repository/20/23.zip>
[16]<https://radimrehurek.com/gensim/>
[17]<https://archive.org/details/enwiki-20170220>
[18]<https://www.mn.uio.no/ifi/english/research/groups/ltg/>
[19]<https://dl.fbaipublicfiles.com/fasttext/vectors-wiki/wiki.en.vec>

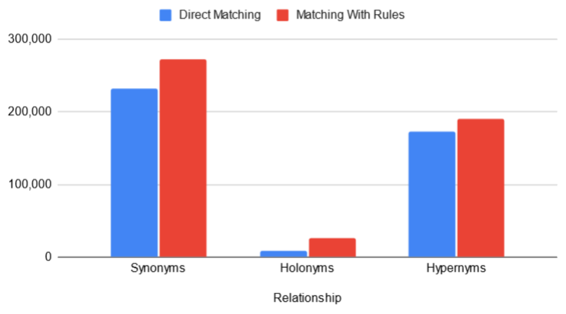
obtained using a model trained on Wikipedia using FastText[20]. The resulting vectors, in this case, also have 300 dimensions and were obtained using the Skip-gram model described in Bojanowski et al. (2017b), with standard parameters. The datasets generated from these sets of word embeddings will be called Gensim and FastText respectively.

Regarding the matching and normalization rules devised, in Figure 4.3 we can see the matches for both datasets with and without the matching and normalization rules mentioned in Section 4.2.1. We can see that the rules increase the number of matches in each of the datasets.

Figure 4.3: Difference in number of matches with both datasets with and without the matching and normalization rules.



(a) Gensim dataset.



(b) FastText dataset.

Source: The Author

Table 4.1 presents statistics for the pairs of words for each relationship for WordNet, for the dataset generated with the Gensim word embeddings, and for the dataset generated with the FastText word embeddings. These are complemented by Table 4.2 and Figure 4.4, showing the percentages of pairs from each relationship per dataset.

Table 4.1: Number of pairs of words of each relationship per dataset.

Relationship	**WordNet**	**Gensim**	**FastText**
Synonymy	549,388	184,814	272,673
Holonymy	105,360	15,536	26,107
Hypernymy	374,736	143,858	189,543
Total	1,029,484	344,208	488,323

Source: The Author

We can verify from the Tables 4.1 and 4.2 and Figure 4.4 that the FastText dataset has a considerably larger number of word pairs than the Gensim dataset after the word matching. Another interesting result is that the proportion of the relationships among the three comparisons (WordNet and the two datasets) remains similar. Another important issue is that Synonymy and Hypernymy are by far the relationships that have the most word

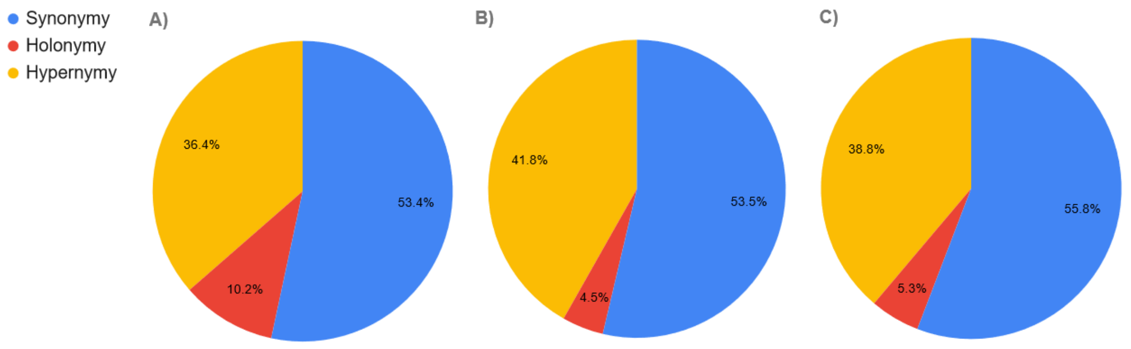
[20]<https://fasttext.cc/>

Table 4.2: Percentage of pairs of words of each relationship per dataset.

Relationship	WordNet	Gensim	FastText
Synonymy	53.37%	53.49%	55.84%
Holonymy	10.23%	4.51%	5.35%
Hypernymy	36.40%	41.79%	38.82%

Source: The Author

Figure 4.4: Percentage of pairs of words of each relationship per dataset: A) WordNet, B) Gensim, C) FastText.



Source: The Author

pairs. It is also interesting to note that even with the FastText dataset, the correspondence of words with WordNet does not reach 50%, showing that it may be possible to find sets of word embeddings that have better correspondence.

Tables 4.3 and 4.4 present statistics for the pairs of words that are multi-labeled for each set of relationships for WordNet, for the dataset generated with the Gensim word embeddings, and for the dataset generated with the FastText word embeddings.

Table 4.3: Number of Pairs of words with more than one relationship per dataset.

Relationships	WordNet	Gensim	FastText
Holonymy, Hypernymy	434	246	270
Holonymy, Hypernymy, Synonymy	87	45	60
Holonymy, Synonymy	7,308	1,852	2,760
Hypernymy, Synonymy	14,728	10,592	12,544
Total	22,557	12,735	15,634

Source: The Author

We can see in the results from Tables 4.3 and 4.4 that the percentage of pairs with multiple relationships for WordNet and the two datasets is less than 4%. Thus, the resulting dataset is multi-labeled, but the percentage of samples with multiple classes is very small. Due to this, and in an effort to deal with a simplified version of the problem, we removed pairs classified by multiple types of relationships. In future works, it's possible

Table 4.4: Percentage of pairs of words with more than one relationship per dataset.

Relationships	WordNet	Gensim	FastText
Holonymy, Hypernymy	0.04%	0.07%	0.06%
Holonymy, Hypernymy, Synonymy	0.01%	0.01%	0.01%
Holonymy, Synonymy	0.71%	0.54%	0.57%
Hypernymy, Synonymy	1.43%	3.08%	2.57%
Total	2.19%	3.70%	3.20%

Source: The Author

to investigate how to handle the original multi-label classification problem.

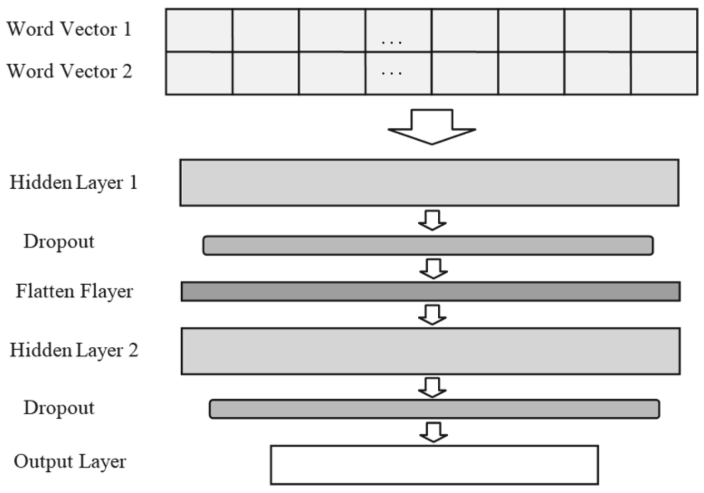
4.3 Approach for classifying semantic relationships

In this work, our focus is on classifying semantic relationships between concepts based on the word embeddings of the words that linguistically represent them. We decided to adopt an approach based on neural networks to perform this classification task. The neural network architecture adopted in this work is the same proposed in Khadir, Guessoum and Aliane (2021). It takes as input two word embedding vectors of 300 dimensions of related words, as a 2×300 matrix. The input layer is followed by one dense layer (called hidden layer in Figure 4.5) with 512 nodes, with ReLU function. It's followed by a dropout of 0.2, in order to mitigate overfitting, and a flatten layer to adjust the data to a unidimensional vector. After these layers, the architecture includes a dense layer with 512 nodes, also with ReLU function, followed by a dropout of 0.2. After that, we have the output layer, which uses the Softmax function and its number of nodes is equal to the number of relationships in the given experiment.

We used the RMSprop optimizer and the categorical cross-entropy loss function for training, except for the experiment using the focal loss function. For the RMSprop optimizer, we use exponential decay with an initial learning rate of 0.001, a decay rate of 0.95, and the number of steps for the decay as 1600. These parameters were decided by experimentation. The architecture can be seen in Figure 4.5.

It is also important to note that for the binary classification experiments we tried using a sigmoid function in the output layer and using binary cross-entropy as the loss function, but, since the performance was equivalent, we decided to keep a single architecture adopting softmax at the output layer.

Figure 4.5: Neural network model used for the classification task.



Source: Khadir, Guessoum and Aliane (2021)

4.4 Experiments

We performed two categories of experiments, one using the complete dataset mentioned in Section 4.2.2, and another one using only the pairs of words related by the hypernymy and holonymy relationships, removing the synonymy relationship from the complete dataset. We decided to remove the synonym pairs because they were the main source of confusion in classifications when analyzing the confusion matrices generated by our tests.

For both categories of experiments, we conducted training with each of the datasets generated by the different word embedding models. We adopted a stratified K-Fold cross-validation process with 5 folds, where 10% of the training data was used for validation. Also, we used 60 epochs for training. We also adopted early stopping for preventing over-fitting, by monitoring the loss of the validation (making almost all experiments end before the 20th epoch). For each category, we adopted both datasets presented in Section 4.2.2, and performed the following list of experiments:

- Baseline experiment.
  - An experiment using the default architecture mentioned in Section 4.3, without applying any approach for dealing with data imbalance.
- Experiments changing the loss function.

  - Using a weighted loss function, using the "compute_class_weight" function from the sklearn library, with the "class_weight" parameter as "balanced", which estimates class weights for imbalanced datasets.
  - Using a focal loss function with $\gamma = 1$ and using "compute_class_weight" with the same parameters used for weighted loss.
- Experiments applying under-sampling.
  - Using random under-sampling with two different variations. For the first one, the majority classes were under-sampled to have the same number of samples as the minority class, and for the second one the majority classes were under-sampled to have twice as many instances as the minority class. This was done because to make the number of instances equal to the majority classes, those classes had to lose a vast number of instances.
  - Using Tomek links under-sampling, under-sampling the majority classes.
- Experiments with over-sampling, with three variations of SMOTE.
  - For the first variation, SMOTE was used to over-sample the minority classes using the 5 nearest neighbors of an instance to define the neighborhood to generate the synthetic samples.
  - For the second variation, we used SMOTE with the same parameters used for the previous variation and then under-sampled the classes with Tomek links.
  - For the last variation, the Borderline SMOTE algorithm was used, over-sampling the majority classes using 5 as the number of nearest neighbors used to define the neighborhood of samples to generate the synthetic samples and 10 as the number of nearest neighbors used to determine if a minority sample is in danger.

For evaluating all experiments, considering that the datasets were imbalanced, we decided to collect macro averages of F1 scores, precision, and recall. For the category of experiments that excluded the synonym pairs we also collected separately the F1, precision, and recall metrics for the holonymy class, which was considered the positive class, since it is the minority class in our setting. It is important to note that we decided to collect the macro averages of the metrics, as weighted and micro averaging can be misleading on account of the classes being highly imbalanced, as can be seen in Table 4.2.

5 RESULTS

In this chapter, we present the results of our experiments. For each experiment we present the F1 scores, precision and recall metrics, also showing the average confusion matrix of the 5 folds of testing. In the confusion matrices, the percentages show the ratio of instances with a certain predicted label over the true label of the instances.

This chapter is divided into the following sections. Section 5.1 presents the experiments to mitigate data imbalance for datasets that classified the pairs into three classes: hypernymy, holonymy and synonymy. Section 5.2 presents the experiments that removed the synonym pairs.

5.1 Experiments with three relationships

In this section, we will present the results of the experiments using the chosen data imbalance mitigation techniques for the complete dataset, with three relationships. Section 5.1.1 presents the results from the baseline experiments, without any method to mitigate data imbalance. Section 5.1.2 presents the results from the loss changes experiments, with weighted loss and focal loss. Section 5.1.3 presents the results from the under-sampling experiments, with random under-sampling and Tomek links under-sampling. Section 5.1.4 presents the results from the over-sampling experiments, with SMOTE, SMOTE combined with Tomek links, and Borderline SMOTE. Section 5.1.5 presents a comparison of the results from the different experiments.

5.1.1 Baseline

As shown in Table 5.1 we achieved a macro F1 score of 0.834 for our baseline experiment using the Gensim dataset, and achieved a macro F1 score of 0.861 for the FastText dataset. Note that in these results with macro averaging recall is smaller than precision, reaching 0.820 compared to 0.850, for Gensim, and 0.851 compared to 0.873, for FastText.

We can see in the confusion matrix shown in Figures 5.1a and 5.1b that one aspect that worsens the macro recall, and consequently the macro F1 score, are the incorrect classifications of holonyms, with most of the incorrect classifications being classified as

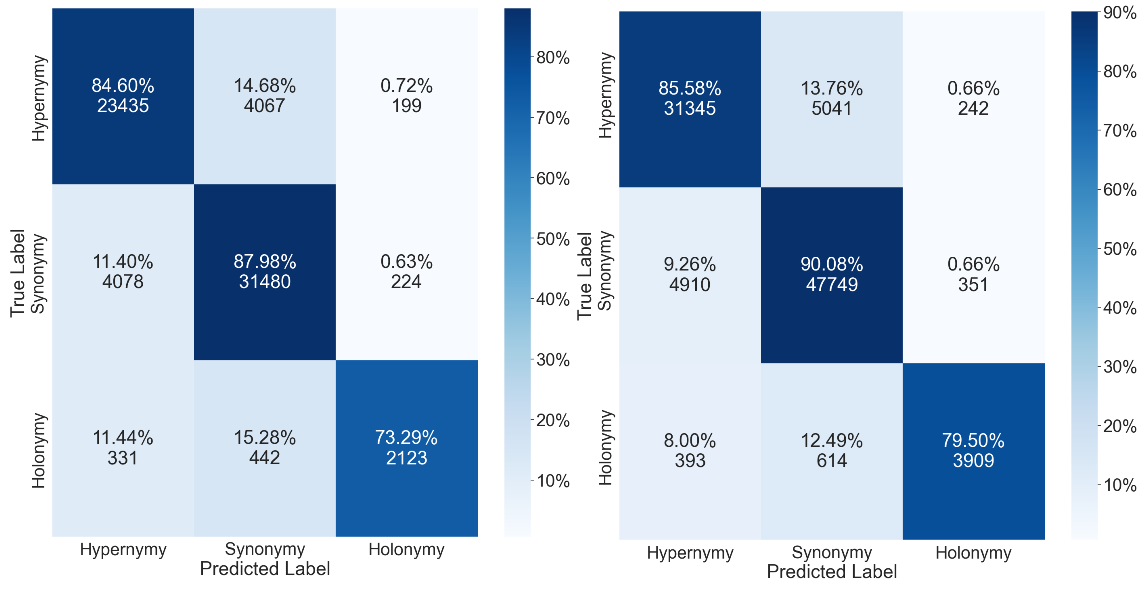
Table 5.1: Evaluation metrics for the baseline model with three relationships.

Dataset	**Macro**		
	F1	**Precision**	**Recall**
Gensim	0.834	0.850	0.820
FastText	0.861	0.873	0.851

Source: The Author

synonyms. For synonyms, most incorrect classifications end up being classified as hypernyms, and for hypernyms most incorrect classifications were classified as synonyms. All the experiments using three relations follow this pattern. It is also worth noting that comparing the holonymy class to the other classes, there is a difference of at least 0.11 for Gensim and at least 0.06 for FastText with the recall of those other two classes.

Figure 5.1: Average confusion matrices for baseline model with three relationships.



(a) Gensim dataset.

(b) FastText dataset.

Source: The Author

5.1.2 Experiments with weighted loss and focal loss

The first experiments to mitigate the data imbalance try to achieve better results by changing the calculation of the loss function. Unfortunately as can be seen in Tables 5.2 and 5.3, the results are consistently worse than the baseline. Considering macro-averaging for the F1 score, compared to the baseline results, there is a decrease of approximately 0.04 with both word embedding datasets with weighted loss. With focal loss we have an ever higher decrease of 0.11 for Gensim and of 0.14 for FastText compared to the baseline results. Given that the weighted loss results were considerably better than the results from the focal loss experiments with both datasets, it would be interesting for future works to investigate if better parameter values could be used for it.

Table 5.2: Evaluation metrics for loss changes for Gensim dataset with three relationships.

Model	**Macro**		
	F1	**Precision**	**Recall**
Weighted	0.791	0.765	0.827
Focal	0.722	0.688	0.806

Source: The Author

Table 5.3: Evaluation metrics for loss changes for FastText dataset with three relationships.

Model	**Macro**		
	F1	**Precision**	**Recall**
Weighted	0.812	0.790	0.842
Focal	0.716	0.683	0.816

Source: The Author

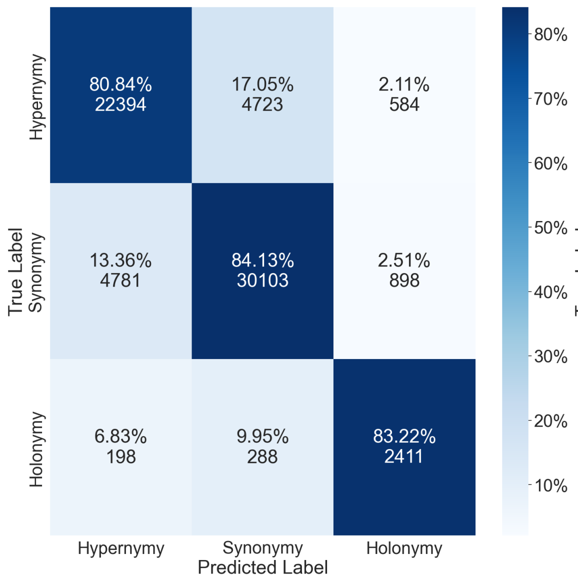
We see in Figures 5.2 and 5.3 that weighted loss greatly improves the recall of the holonymy class compared to the baseline, with an increase of 0.09 for Gensim and 0.05 for FastText. However, weighted loss ends up worsening the recall of the other classes compared to the baseline, with a decrease of approximately 0.04 for the other classes with both datasets. With focal loss, compared to weighted loss, there was an increase in the recall of holonyms by 0.04 for Gensim and 0.05 for FastText, but the recall of the other classes was worsened, for the hypernymy class we have a decrease of 0.04 for Gensim and 0.03 for FastText, and for the synonymy class we have a decrease of 0.05 for Gensim and 0.10 for FastText.

5.1.3 Under-sampling experiments.

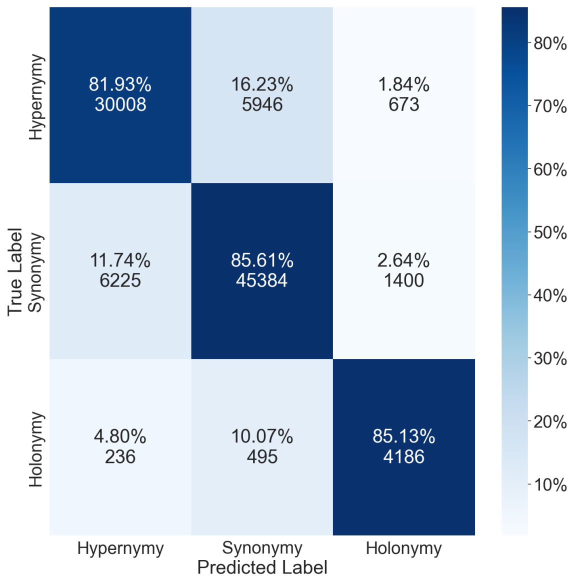
The second set of experiments tries to mitigate data imbalance by using under-sampling techniques. In the results, we reference the variations described in Section 4.4 as "random", for the first variation of random under-sampling, and as "random 2", for the second variation.

The results from the random under-sampling technique in Tables 5.4 and 5.5 show that they worsened the model significantly compared to the baseline. Considering macro F1 score, there was a decrease of 0.08 to 0.17 across the variations and datasets. Even

Figure 5.2: Average confusion matrices for weighted loss experiments with three relationships.



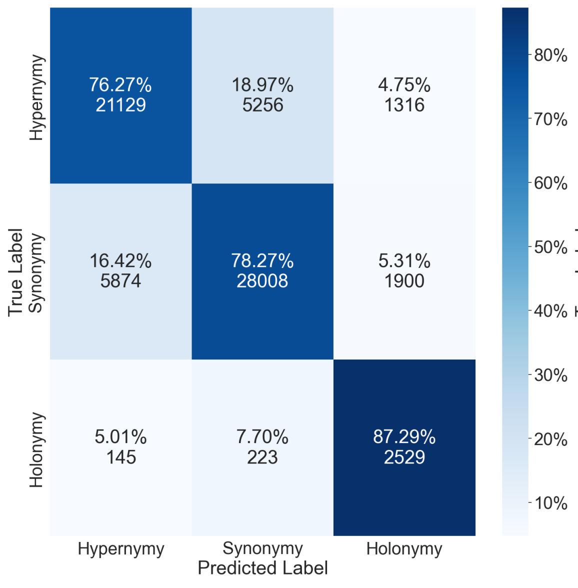
(a) Gensim dataset.



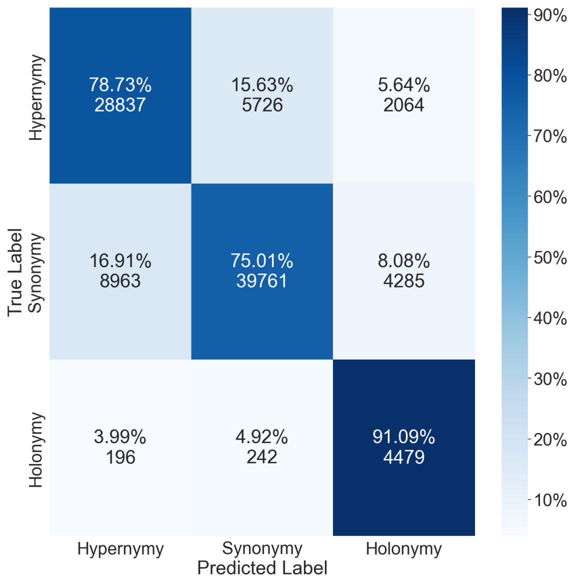
(b) FastText dataset.

Source: The Author

Figure 5.3: Average confusion matrices for focal loss experiments with three relationships.



(a) Gensim dataset.



(b) FastText dataset.

Source: The Author

though doubling the number of instances of the majority classes increased F1 scores by 0.06 for Gensim and 0.05 for FastText, compared to using random under-sampling to equal the number of instances per class, the results still were inferior to the baseline model. The results from the Tomek links experiment are the closest we got to the baseline model using under-sampling techniques, but they still do not surpass the baseline results for the macro F1 score.

Table 5.4: Evaluation metrics for under-sampling for Gensim dataset with three relationships.

Model	**Macro**		
	F1	**Precision**	**Recall**
Random	0.661	0.632	0.745
Random 2	0.725	0.697	0.776
Tomek	0.811	0.831	0.795

Source: The Author

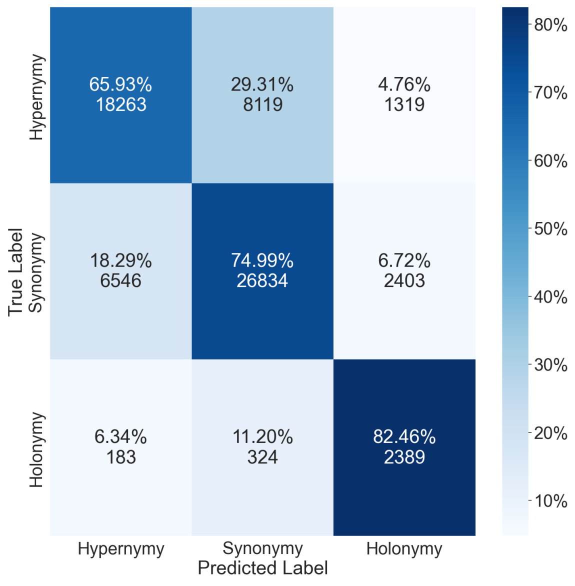
Table 5.5: Evaluation metrics for under-sampling for FastText dataset with three relationships.

Model	**Macro**		
	F1	**Precision**	**Recall**
Random	0.722	0.686	0.798
Random 2	0.777	0.753	0.813
Tomek	0.852	0.859	0.845

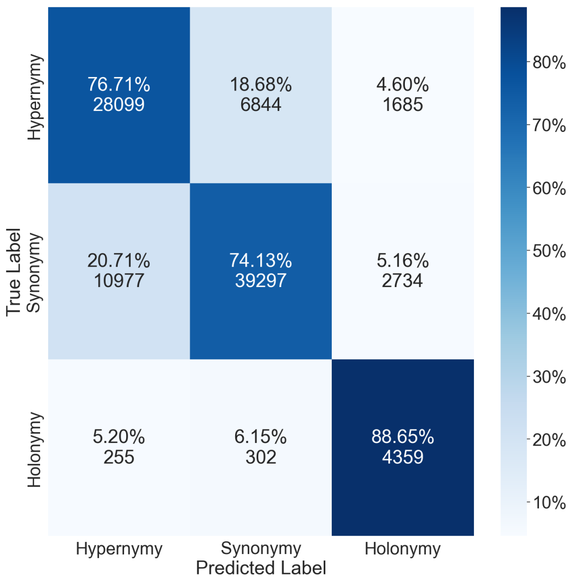
Source: The Author

Considering the confusion matrices shown in Figures 5.4, 5.5 and 5.6, for random under-sampling methods, compared to baseline results, we see an improvement in the recall of holonyms, with an increase of approximately 0.09 for the first variation, and of 0.07 for Gensim and of 0.05 for FastText for the second variation. However, there was a decrease in the recall of other classes, with a decrease for the synonymy class of at least 0.09 and for hypernyms of at least 0.06 across all variations and datasets. When comparing the second random under-sampling variation with the first, there is an improvement in the recall of the hypernymy and synonymy class, but there is a deterioration in the recall of holonyms. Using Tomek Links, we see a slight deterioration in the recall of synonyms and hypernyms compared to the baseline. For the holonymy class, we also see a similar recall to the baseline, 0.700 compared to 0.732 from the baseline for Gensim, 0.802 compared to 0.795 from the baseline for FastText. Considering those results, Tomek links perform considerably better than random under-sampling, unfortunately, the outcome is still worse with this method when comparing it to the baseline results.

Figure 5.4: Average confusion matrices for random under-sampling experiments with three relationships.



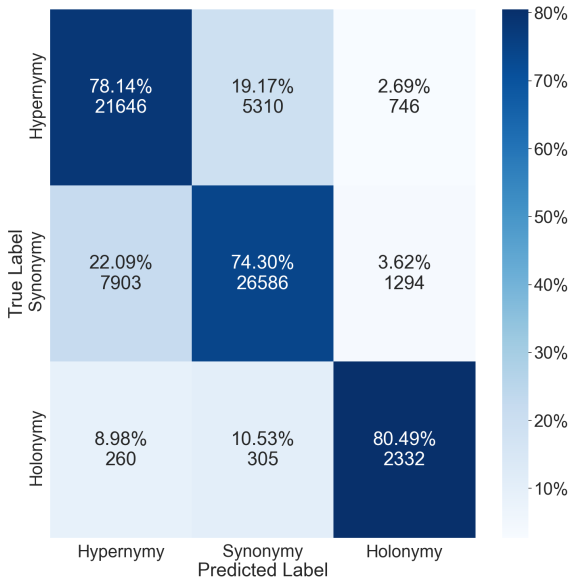
(a) Gensim dataset.



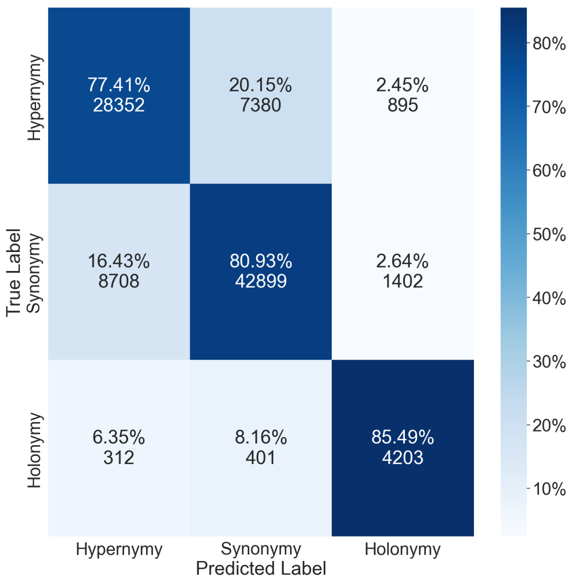
(b) FastText dataset.

Source: The Author

Figure 5.5: Average confusion matrices for doubled random under-sampling experiments with three relationships.



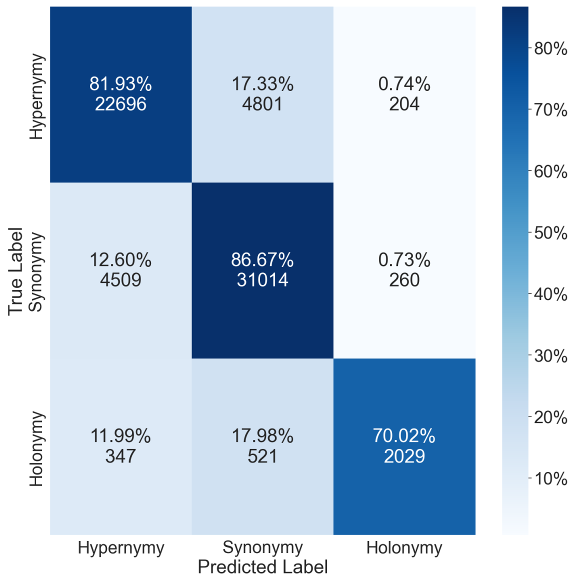
(a) Gensim dataset.



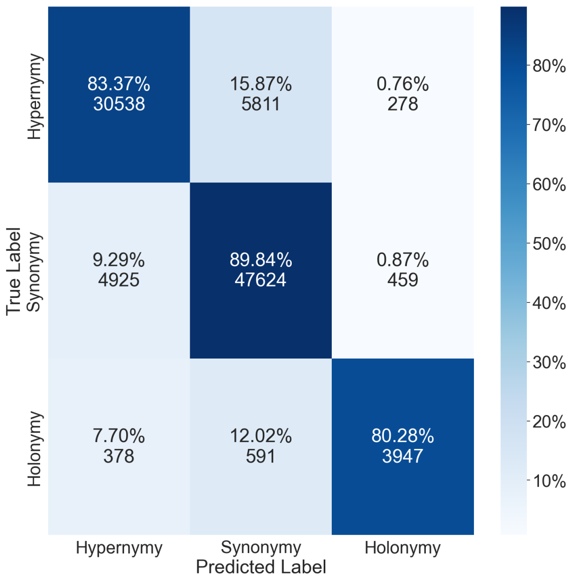
(b) FastText dataset.

Source: The Author

Figure 5.6: Average confusion matrices for Tomek links experiments with three relationships.



(a) Gensim dataset.



(b) FastText dataset.

Source: The Author

5.1.4 Over-sampling experiments

For over-sampling, we tried using three variations of SMOTE presented in Section 4.4. The results of these experiments are presented in Tables 5.6 and 5.7. Considering macro F1 scores, The three SMOTE variations respectively achieved an F1 score of 0.830, 0.819 and 0.830 compared to 0.834 from the baseline for the Gensim dataset, and achieved an F1 score of 0.866, 0.861 and 0.866 compared to 0.861 from the baseline for the FastText dataset. All variations got close to the baseline model, or slightly surpassed it using the FastText dataset when using SMOTE and SMOTE Borderline.

Table 5.6: Evaluation metrics for over-sampling for Gensim dataset with three relationships.

Model	**Macro**		
	F1	**Precision**	**Recall**
SMOTE	0.830	0.831	0.828
SMOTE/Tomek	0.819	0.816	0.822
SMOTE Borderline	0.830	0.831	0.830

Source: The Author

Analyzing the three average confusion matrices for the SMOTE variations shown in Figures 5.7, 5.8 and 5.9, we see a significant improvement in the holonymy class recall compared to the baseline, with an increase of approximately 0.05 for both datasets. And

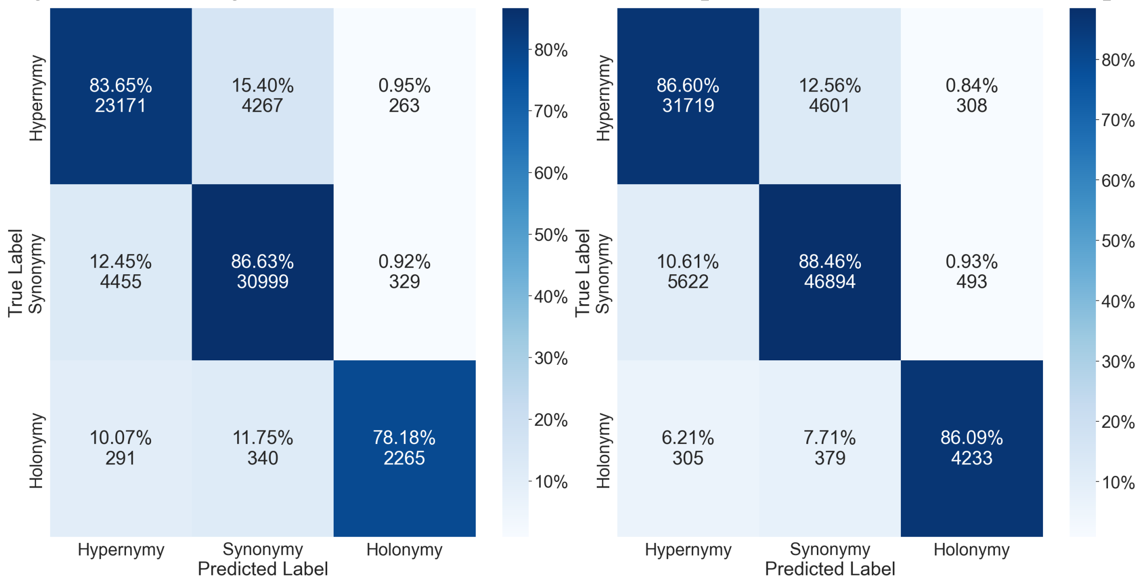
Table 5.7: Evaluation metrics for over-sampling for FastText dataset with three relationships.

Model	Macro		
	F1	Precision	Recall
SMOTE	0.866	0.863	0.870
SMOTE/Tomek	0.861	0.856	0.866
SMOTE Borderline	0.866	0.864	0.869

Source: The Author

with a slight deterioration in the recall of other classes, with a decrease of 0.01 to 0.03 across datasets for the synonymy class, and varying from decreasing of 0.01 to increasing of 0.01 for the hypernymy class.

Figure 5.7: Average confusion matrices for SMOTE experiments with three relationships.



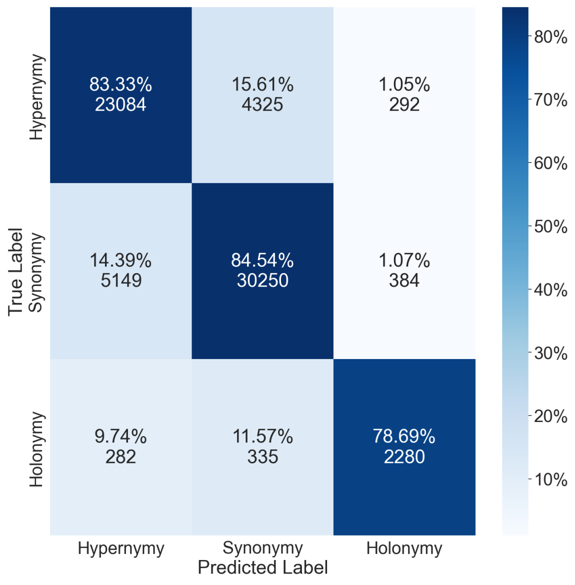
(a) Gensim dataset. (b) FastText dataset.

Source: The Author

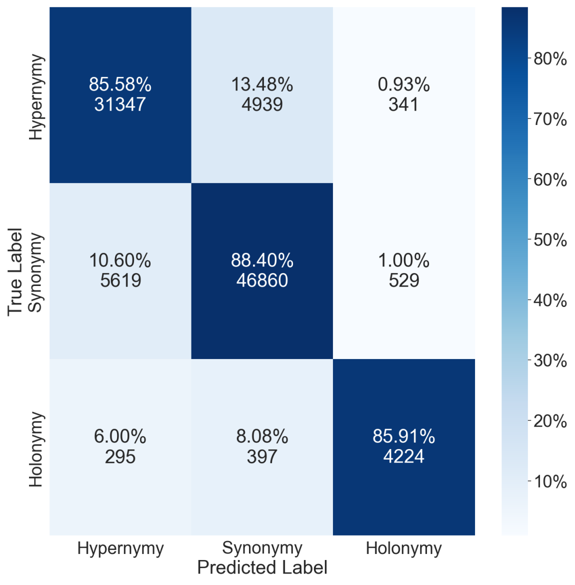
5.1.5 Comparing results

The results of the experiments are summarized in Tables 5.8 and 5.9. Considering the macro F1 score from all experiments, we can see that they showed worse results than the baseline model. There were only slight improvements in other metrics, such as the random under-sampling methods improving macro recall of the holonymy class, but worsening it for other classes. It is important to notice that even with some improvements in recall of classes in some experiments, all experiments showed the same pattern seen in the baseline classification task.

Figure 5.8: Average confusion matrices for SMOTE/Tomek experiments with three relationships.



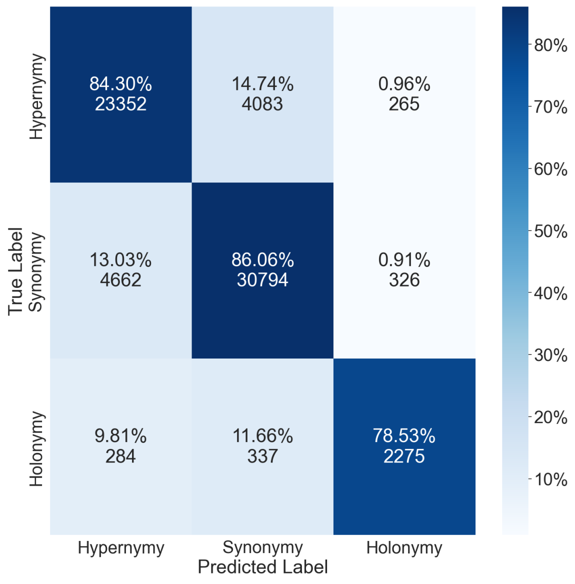
(a) Gensim dataset.



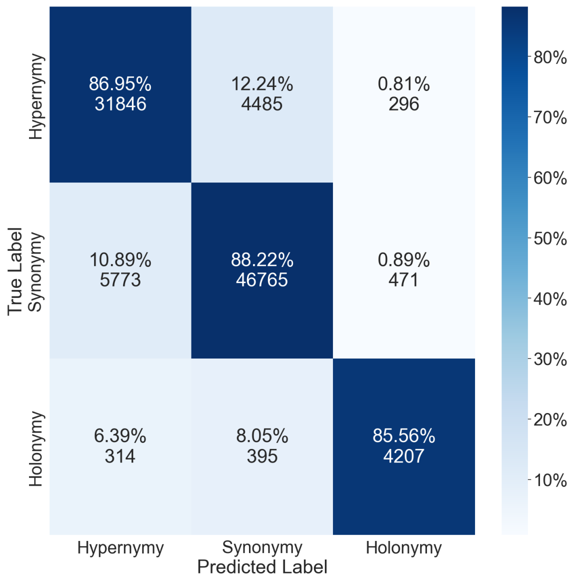
(b) FastText dataset.

Source: The Author

Figure 5.9: Average confusion matrices for Borderline SMOTE experiments with three relationships.



(a) Gensim dataset.



(b) FastText dataset.

Source: The Author

The most interesting results are from the Tomek links experiments and SMOTE experiments. Tomek links and its combination with SMOTE resulted in similar results to the baseline model, but with the added training time for running these algorithms, and not considerable improvements. Overall, the over-sampling methods SMOTE and Borderline SMOTE have similar results, slightly superior for the FastText dataset, when we compare them with baseline results. However, when we consider that the two methods significantly improve the recall of the holonymy class, by approximately 0.05, with the tradeoff of slightly worsening it for other classes, these models prove to be interesting choices if having a better recall for that class is a desired result.

Table 5.8: Evaluation metrics for experiments for Gensim dataset with three relationships.

Model	**Macro**		
	F1	**Precision**	**Recall**
Baseline	0.834	0.850	0.820
Weighted	0.791	0.765	0.827
Focal	0.722	0.688	0.806
Random	0.661	0.632	0.745
Random 2	0.725	0.697	0.776
Tomek	0.811	0.831	0.795
SMOTE	0.830	0.831	0.828
SMOTE/Tomek	0.819	0.816	0.822
SMOTE Borderline	0.830	0.831	0.830

Source: The Author

Table 5.9: Evaluation metrics for experiments for FastText dataset with three relationships.

Model	**Macro**		
	F1	**Precision**	**Recall**
Baseline	0.861	0.873	0.851
Weighted	0.812	0.790	0.842
Focal	0.716	0.683	0.816
Random	0.722	0.686	0.798
Random 2	0.777	0.753	0.813
Tomek	0.852	0.859	0.845
SMOTE	0.866	0.863	0.870
SMOTE/Tomek	0.861	0.856	0.866
SMOTE Borderline	0.866	0.864	0.869

Source: The Author

5.2 Experiments with two relationships

After analyzing the previous results, we saw that many of the errors in the confusion matrices of all experiments were related to wrong classifications of synonyms, or pairs from other classes being classified as synonyms. With that in mind, we decided to perform experiments without considering the synonym relationship. Since we only have two possible classes in the classification task, we chose the holonymy class as the positive class, as that is the minority class, representing less than 0.15 of the dataset. It is also worth noting that for the results, we are analyzing mainly the F1 score, recall, and precision metrics considering the positive class, but we calculated the macro averaged metrics to compare them with the experiments with three relationships.

This section is divided as follows. Section 5.2.1 presents the results from the baseline experiments, without any method to mitigate data imbalance. Section 5.2.2 presents the results from the loss changes experiments, with weighted loss and focal loss. Section 5.2.3 presents the results from the under-sampling experiments, with random under-sampling and Tomek links under-sampling. Section 5.2.4 presents the results from the over-sampling experiments, with SMOTE, SMOTE combined with Tomek links, and Borderline SMOTE. Section 5.2.5 presents a comparison of the results from the different experiments., comparing them with the results from the experiments with three relationships as well.

5.2.1 Baseline

As shown in Table 5.10 we achieved an F1 score of 0.828 for our baseline experiment using the Gensim dataset and achieved an F1 score of 0.885 for the FastText dataset. Similar to the baseline model for three relationships, recall is smaller than precision, 0.772 compared to 0.893 for Gensim and 0.848 compared to 0.925 for FastText, bringing the F1 score down.

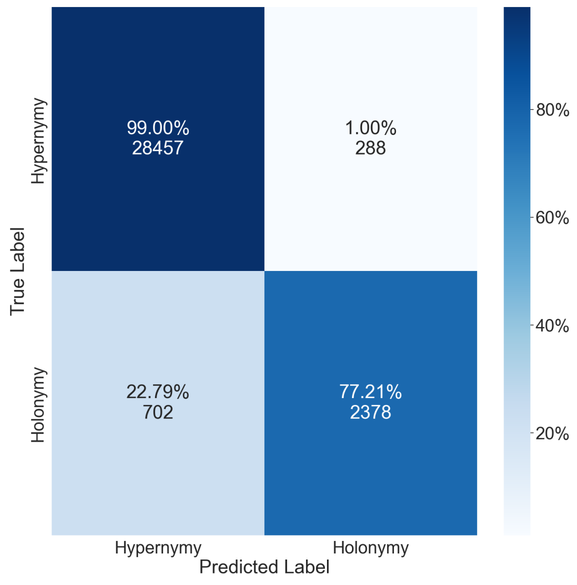
Table 5.10: Evaluation metrics for baseline model with two relationships.

Dataset	**Positive**			**Macro**		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
Gensim	0.828	0.893	0.772	0.905	0.934	0.881
FastText	0.885	0.925	0.848	0.934	0.952	0.919

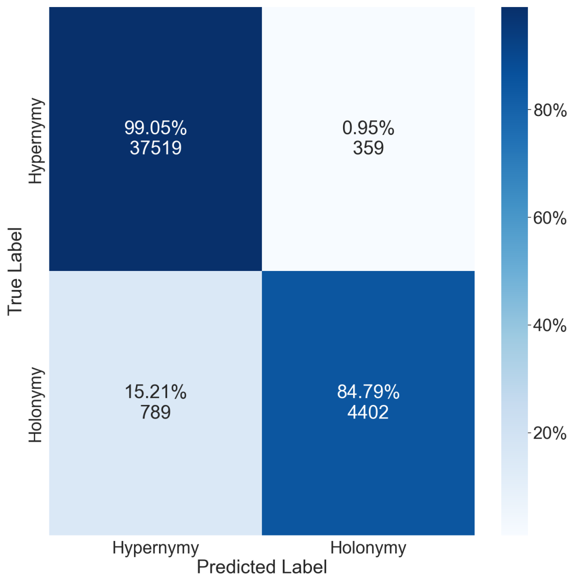
Source: The Author

We can see in the confusion matrix shown in Figures 5.10a and 5.10b that incorrect classifications of hypernyms are fairly low, with the class having a recall of 0.990 for both datasets.

Figure 5.10: Average confusion matrices for baseline model with two relationships.



(a) Gensim dataset.



(b) FastText dataset.

Source: The Author

5.2.2 Weighted loss and focal loss experiments

The results from the changes to the loss function experiments can be seen in Tables 5.11 and 5.12. Similar to the experiments performed with three relationships, the results are worse than the baseline results, with a decrease of 0.04 for Gensim and 0.02 for FastText for weighted loss, and of 0.10 for focal loss. It's worth noting that similar to the experiments with three relationships, the weighted loss metrics were better than the focal loss metrics. In both changes, recall is superior to precision, different from the baseline for two relationships, with a difference between the metrics of 0.16 for Gensim and 0.10 for FastText for weighted loss, and for focal loss with a difference between the metrics of 0.34 for Gensim and 0.26 for FastText.

We see in Figures 5.11 and 5.12 that, compared to the baseline results, weighted loss greatly increases recall for the holonymy class, an increase of 0.10 for Gensim and of 0.06 for FastText. But, it decreases recall for the hypernymy class, by approximately 0.02 for both datasets, making the tradeoff not worth it overall and resulting in decreased preci-

Table 5.11: Evaluation metrics for loss changes for Gensim dataset with two relationships.

Model	Positive			Macro		
	F1	Precision	Recall	F1	Precision	Recall
Weighted	0.783	0.709	0.877	0.878	0.847	0.918
Focal	0.719	0.588	0.930	0.839	0.790	0.929

Source: The Author

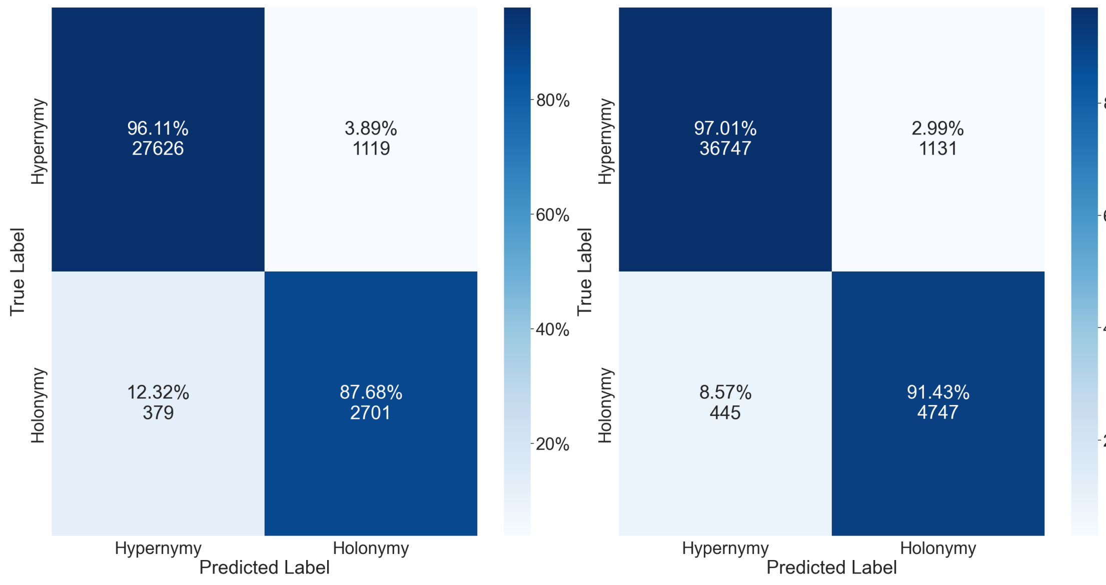
Table 5.12: Evaluation metrics for loss changes for FastText dataset with two relationships.

Model	Positive			Macro		
	F1	Precision	Recall	F1	Precision	Recall
Weighted	0.858	0.809	0.914	0.918	0.898	0.942
Focal	0.785	0.675	0.943	0.873	0.833	0.939

Source: The Author

sion. Focal loss, also increases the recall of the holonymy class, even more than weighted loss, 0.05 for Gensim and 0.02 for FastText compared to weighted loss results, but decreases recall for the hypernymy class even more, approximately 0.03 for both datasets compared to weighted loss results, having even a bigger disparity between precision and recall.

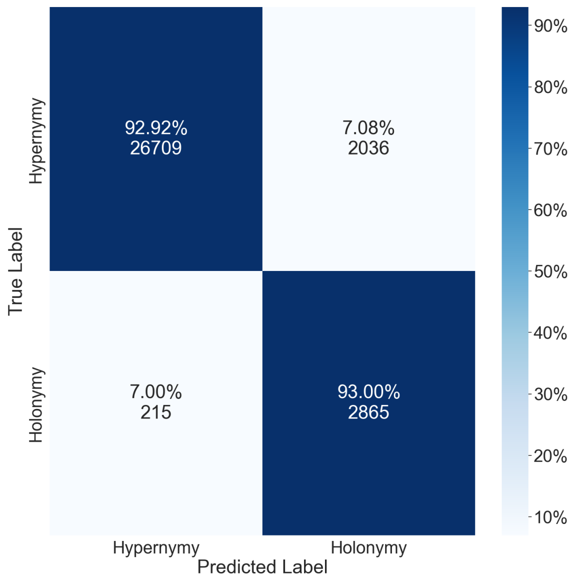
Figure 5.11: Average confusion matrices for weighted loss experiments with two relationships.



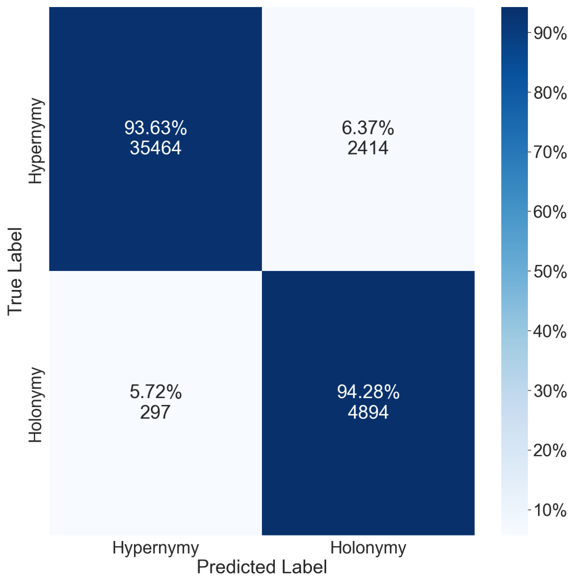
(a) Gensim dataset. (b) FastText dataset.

Source: The Author

Figure 5.12: Average confusion matrices for focal loss experiments with two relationships.



(a) Gensim dataset.



(b) FastText dataset.

Source: The Author

5.2.3 Under-sampling experiments.

For the under-sampling experiments, the results can be seen in Tables 5.13 and 5.14. In the results, we reference the variations described in Section 4.4 as "random", for the first variation of random under-sampling, and as "random 2", for the second variation. The random under-sampling results show that both variations worsened the model significantly, with a decrease in F1 score for the first variation of 0.11 and 0.09 for Gensim and FastText respectively, and a decrease of approximately 0.05 for the second variation. Similar to the experiments with three relationships, the second variation achieved a better F1 score when compared to the first. Using Tomek links under-sampling, we managed to have a similar F1 score to the baseline model with two relationships with an F1 score of 0.832 for Gensim and 0.886 for FastText.

Table 5.13: Evaluation metrics for under-sampling for Gensim dataset with two relationships.

Model	Positive			Macro		
	F1	Precision	Recall	F1	Precision	Recall
Random	0.711	0.602	0.876	0.835	0.794	0.906
Random 2	0.763	0.696	0.848	0.867	0.839	0.903
Tomek	0.832	0.878	0.792	0.907	0.927	0.889

Source: The Author

Table 5.14: Evaluation metrics for under-sampling for FastText dataset with two relationships.

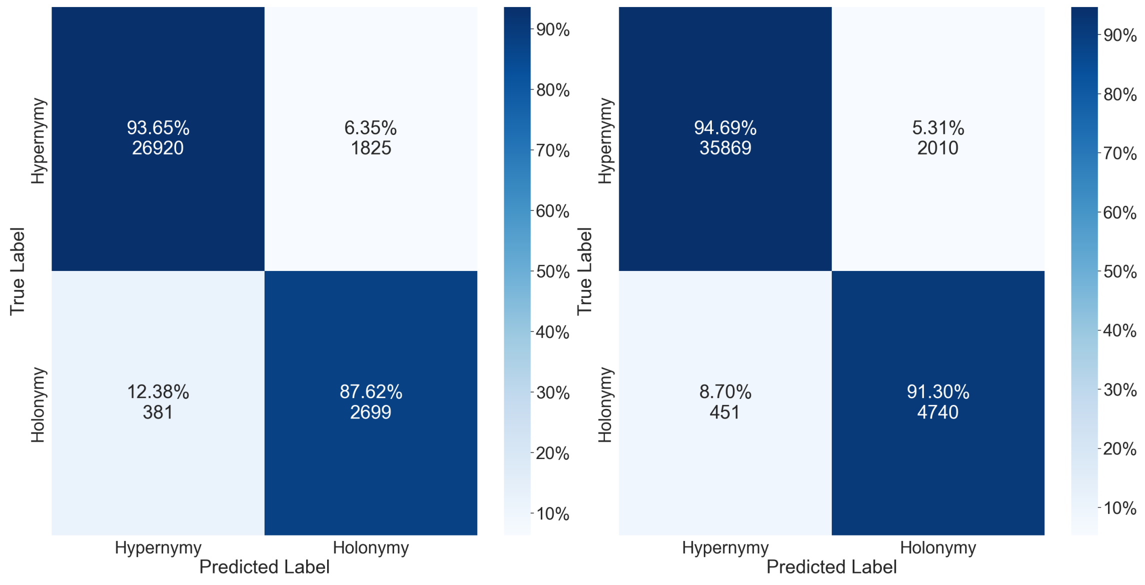
Model	Positive			Macro		
	F1	Precision	Recall	F1	Precision	Recall
Random	0.794	0.705	0.913	0.880	0.846	0.929
Random 2	0.831	0.799	0.866	0.903	0.890	0.917
Tomek	0.886	0.904	0.870	0.935	0.942	0.928

Source: The Author

Considering the confusion matrices shown in Figures 5.13, 5.14 and 5.15, and considering the random variations respectively, the random under-sampling experiments achieved an increase in recall compared to the baseline for the holonymy class of 0.10 and 0.07 for Gensim and of 0.06 and 0.01 for FastText. For the hypernymy class, the results show, compared to the baseline, a decrease in recall of 0.05 and 0.03 for Gensim, and of 0.04 and 0.02 for FastText. Those results show that with the random under-sampling variations, we increase recall for the holonymy class, decreasing recall for the hypernymy class, similar to the loss experiments.

For Tomek links under-sampling we have similar results to the baseline model for two relationships, with a recall for the holonymy class equal to 0.791 for Gensim and 0.869 for FastText, and with a recall for the hypernymy class of 0.988 for Gensim and 0.987 for FastText.

Figure 5.13: Average confusion matrices for random under-sampling experiments with two relationships.

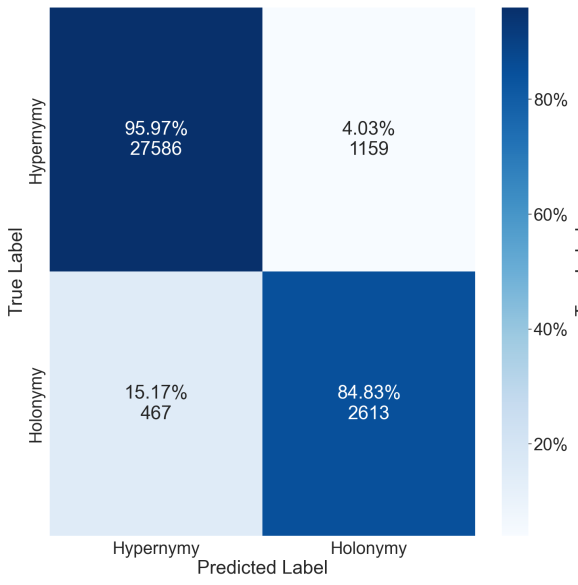


(a) Gensim dataset.

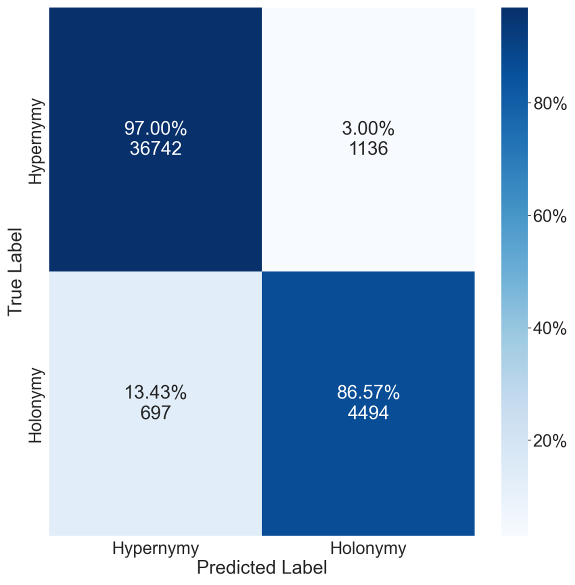
(b) FastText dataset.

Source: The Author

Figure 5.14: Average confusion matrices for doubled random under-sampling experiments with two relationships.



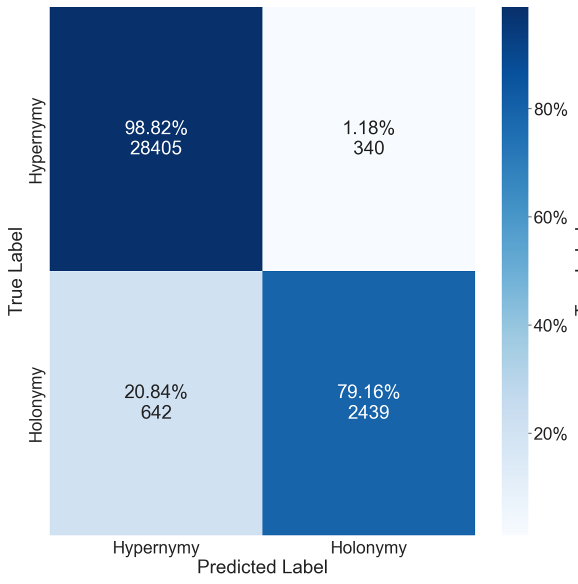
(a) Gensim dataset.



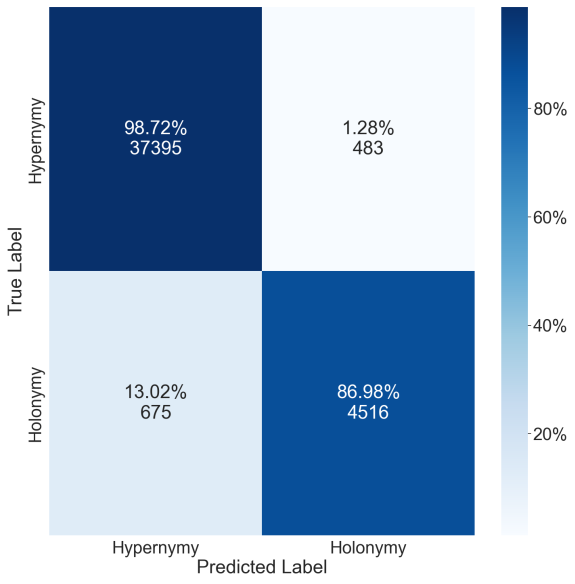
(b) FastText dataset.

Source: The Author

Figure 5.15: Average confusion matrices for Tomek links experiments with two relationships.



(a) Gensim dataset.



(b) FastText dataset.

Source: The Author

5.2.4 Over-sampling experiments

For the over-sampling experiments, the evaluation metrics can be seen in Tables 5.15 and 5.16. All variations got close to the baseline model for two relationships in regards to F1 score, with an F1 score of approximately 0.815 for the three variations compared to 0.828 from the baseline for Gensim, and of 0.877, 0.882 and 0.879 compared to 0.885 from the baseline for FastText. In all variations, recall was better than the baseline model and precision was worse. With an increase in recall of 0.04 to 0.07 across all variations for Gensim, and of 0.02 to 0.04 across all variations for FastText, and with a decrease in precision of 0.08 to 0.10 across all variations for Gensim, and of 0.04 to 0.06 across all variations for FastText.

Table 5.15: Evaluation metrics for over-sampling for Gensim dataset with two relationships.

Model	**Positive**			**Macro**		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
SMOTE	0.815	0.811	0.820	0.897	0.895	0.899
SMOTE/Tomek	0.816	0.792	0.844	0.898	0.887	0.910
SMOTE Borderline	0.816	0.805	0.827	0.897	0.893	0.902

Source: The Author

Table 5.16: Evaluation metrics for over-sampling for FastText dataset with two relationships.

Model	**Positive**			**Macro**		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
SMOTE	0.877	0.878	0.876	0.930	0.930	0.929
SMOTE/Tomek	0.882	0.878	0.886	0.932	0.931	0.934
SMOTE Borderline	0.879	0.865	0.894	0.931	0.925	0.937

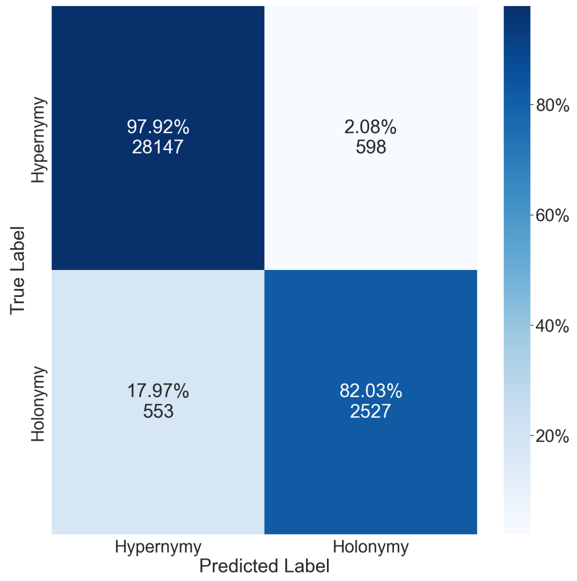
Source: The Author

Analyzing the three average confusion matrices for the SMOTE variations shown in Figures 5.16, 5.17 and 5.18, we have similar results to the baseline model, with an increase of recall for the holonymy class and a slight decrease of recall for the hypernymy class compared to the baseline.

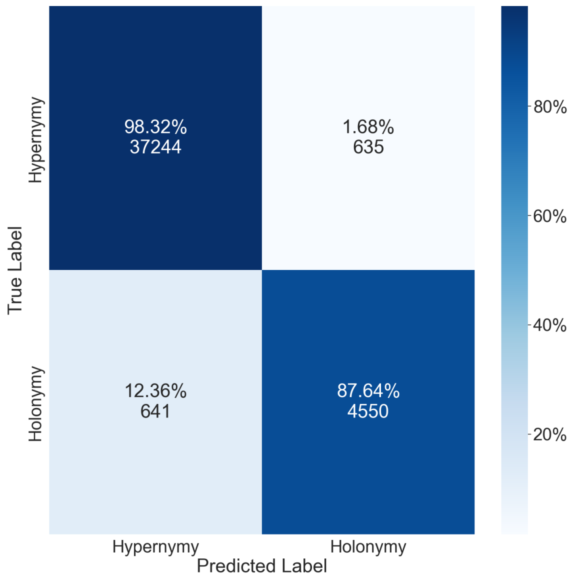
5.2.5 Comparing results

The results of the experiments are summarized in Tables 5.17 and 5.18. It is important to note that the results were found to be extremely similar to the results with the

Figure 5.16: Average confusion matrices for SMOTE experiments with two relationships.



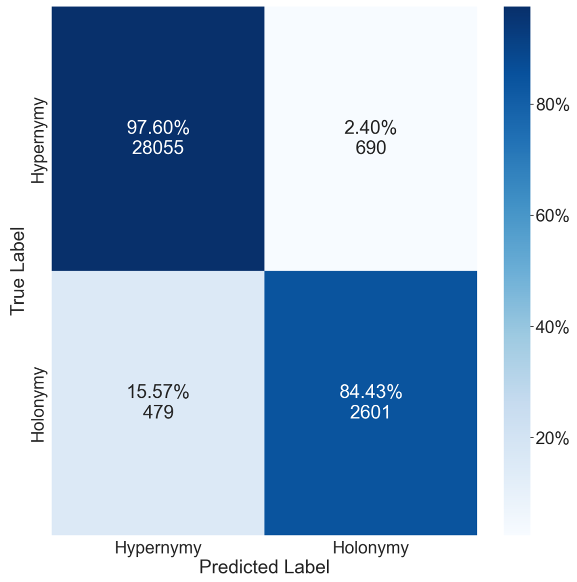
(a) Gensim dataset.



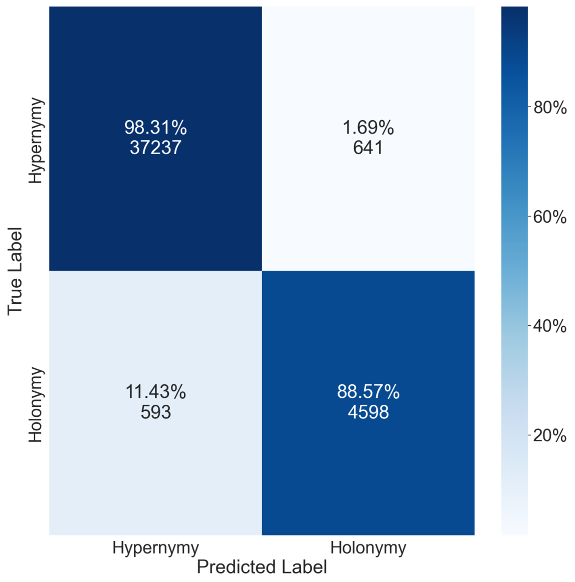
(b) FastText dataset.

Source: The Author

Figure 5.17: Average confusion matrices for SMOTE/Tomek experiments with two relationships.



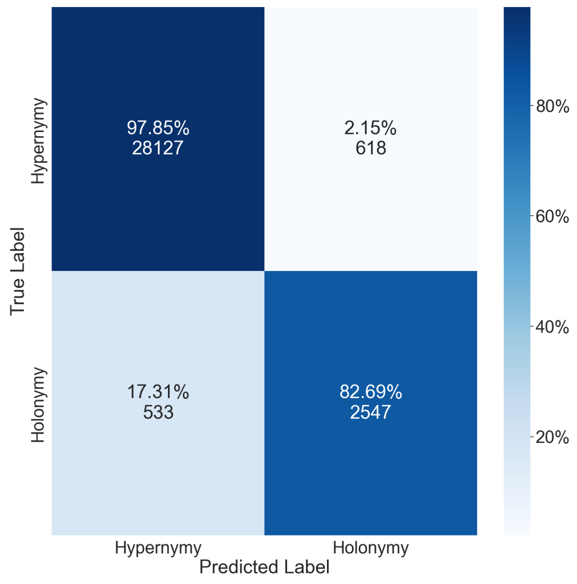
(a) Gensim dataset.



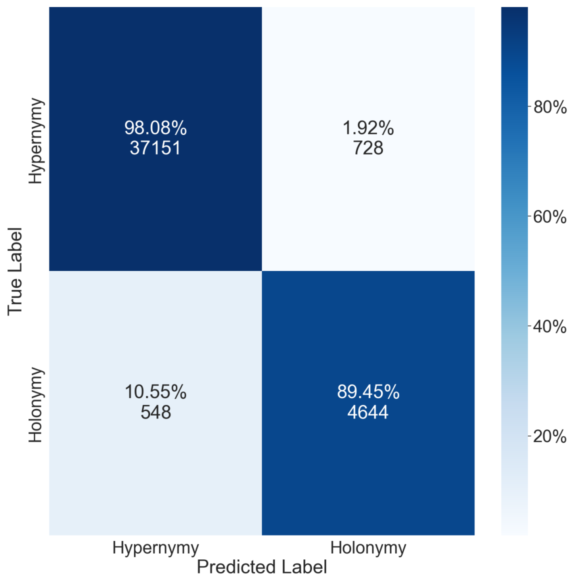
(b) FastText dataset.

Source: The Author

Figure 5.18: Average confusion matrices for Borderline SMOTE experiments with two relationships.



(a) Gensim dataset.



(b) FastText dataset.

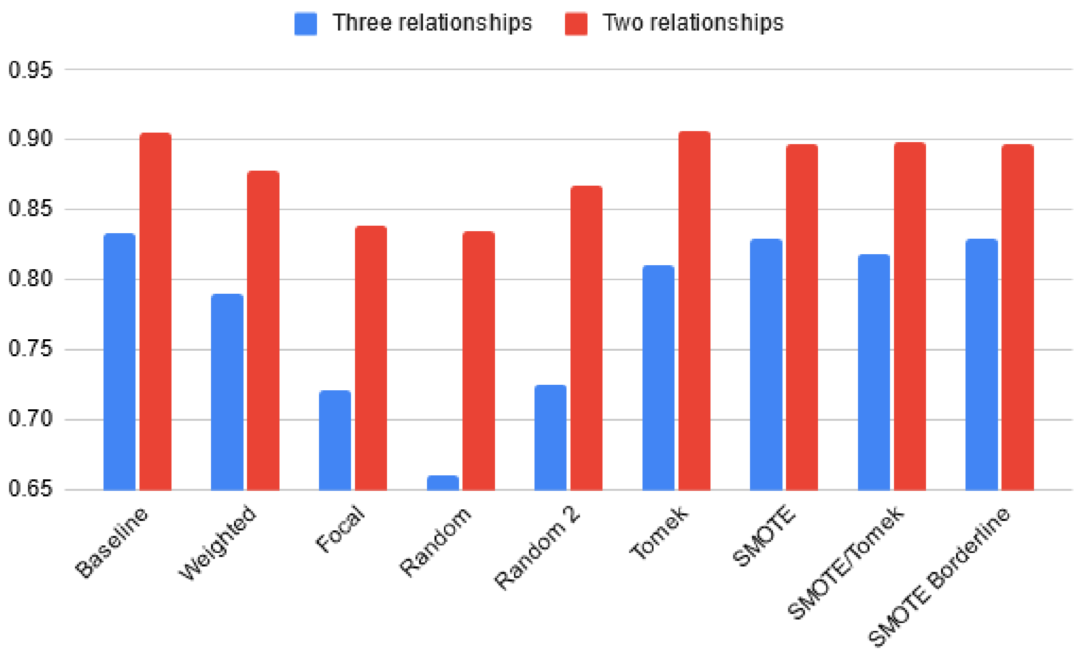
Source: The Author

complete dataset, but with better F1 scores, comparing the macro averages. With an increase of macro F1 score of at least 0.06 when comparing these results to the experiments with the complete dataset. For the baseline model, the increase was approximately 0.07, and for other variations, the increase was as high as 0.17, for random under-sampling for Gensim. That increase indicates that potentially removing relations from the classification task of semantic relations that cause many misclassifications, such as the synonymy relationship in our task, can result in substantial gains in terms of F1 scores. The Macro F1 scores from the experiments with three relationships and two relationships can be seen in Figure 5.19 and Figure 5.20.

One distinction from the results from the complete dataset is that recall and precision did not follow a pattern. In the baseline tests, precision consistently surpassed recall, affecting the F1 score. When changing the loss function, the results were found to be less effective than the baseline, although the weighted loss metrics outperformed the focal loss metrics, but both variations showed a superior recall at the expense of precision. The most significant observation in these experiments was the increase in the holonymy class recall, with a significant decrease in the hypernymy class recall.

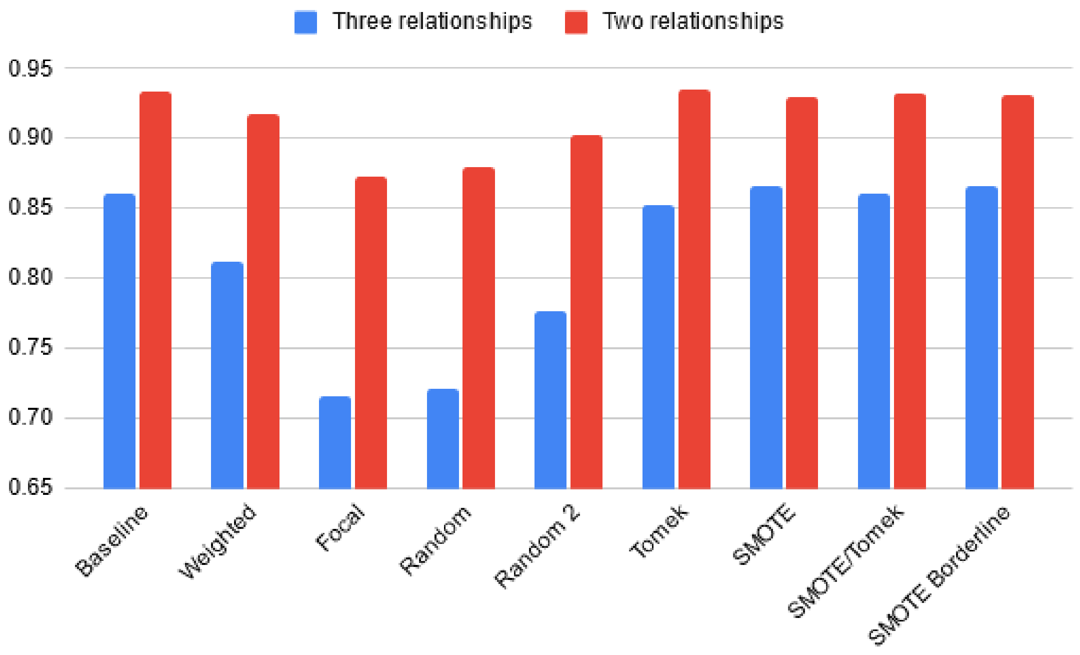
For the under-sampling experiments, random under-sampling had worse results from the baseline, and affected recall similarly to the loss change experiments. However, Tomek links under-sampling managed to achieve an F1 score comparable to the baseline.

Figure 5.19: Macro F1 scores from the experiments with three relationships and two relationships for Gensim.



Source: The Author

Figure 5.20: Macro F1 scores from the experiments with three relationships and two relationships for FastText.



Source: The Author

The over-sampling experiments with SMOTE, presented results that matched the baseline results as well. While these techniques enhanced recall, they did so by decreasing precision. It is interesting to note that from the results from random under-sampling, it can also be concluded that for this problem, not only is the imbalance a challenge, but the amount of data is very relevant. This is because eliminating the imbalance, by drastically reducing the size of the dataset, resulted in worse performance than reducing the imbalance to a lesser extent but retaining more data.

While different experiments presented their own positives and negatives, the experiments with the best results, the SMOTE variations and Tomek links under-sampling, were only able to come close to the baseline performance. With that in mind, the baseline model is still the best model considering these results.

Table 5.17: Evaluation metrics for experiments for Gensim dataset with two relationships.

Model	**Positive**			**Macro**		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
Baseline	0.828	0.893	0.772	0.905	0.934	0.881
Weighted	0.783	0.709	0.877	0.878	0.847	0.918
Focal	0.719	0.588	0.930	0.839	0.790	0.929
Random	0.711	0.602	0.876	0.835	0.794	0.906
Random 2	0.763	0.696	0.848	0.867	0.839	0.903
Tomek	0.832	0.878	0.792	0.907	0.927	0.889
SMOTE	0.815	0.811	0.820	0.897	0.895	0.899
SMOTE/Tomek	0.816	0.792	0.844	0.898	0.887	0.910
SMOTE Borderline	0.816	0.805	0.827	0.897	0.893	0.902

Source: The Author

Table 5.18: Evaluation metrics for experiments for FastText dataset with two relationships.

Model	**Positive**			**Macro**		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
Baseline	0.885	0.925	0.848	0.934	0.952	0.919
Weighted	0.858	0.809	0.914	0.918	0.898	0.942
Focal	0.785	0.675	0.943	0.873	0.833	0.939
Random	0.794	0.705	0.913	0.880	0.846	0.929
Random 2	0.831	0.799	0.866	0.903	0.890	0.917
Tomek	0.886	0.904	0.870	0.935	0.942	0.928
SMOTE	0.877	0.878	0.876	0.930	0.930	0.929
SMOTE/Tomek	0.882	0.878	0.886	0.932	0.931	0.934
SMOTE Borderline	0.879	0.865	0.894	0.931	0.925	0.937

Source: The Author

6 CONCLUSION

In this work, we investigated a machine learning approach for classifying semantic relations from word embeddings that represent concepts. When reviewing the related works, we found that in this scenario, it is common for datasets to be highly imbalanced, and we could not find works that tried to systematically investigate approaches for dealing with this issue in this scenario. Thus, the main goal of this work was to evaluate common techniques to mitigate data imbalance in classification tasks.

Since the literature does not provide datasets for our target task, it was necessary to develop our own datasets for carrying out the evaluation of data imbalance mitigation techniques. Thus, as a second goal, we developed a methodology for building datasets for the task of classifying semantic relations from word embeddings. And, finally, following our methodology, we developed two different datasets, with two variations, that were adopted in our experiments.

The methodology proposed in this work for building datasets for the target task considers as input a set of pre-trained word embeddings and adopts the WordNet lexical resource as a reference for establishing semantic relationships between pairs of words in the word embeddings set. In order to do that, we defined a set of normalization rules and matching rules, to be used during the execution of the matching process for mapping the words in the word embeddings set and the words specified in WordNet.

We applied our proposed methodology to developing two different datasets (with two variations). In order to do that, firstly, we selected two different sets of pre-trained static word embedding, which we named Gensim and FastText, with different characteristics. After that, we applied our methodology to these two sets of word embeddings for building two datasets suitable for being used as input for machine learning approaches. We developed two versions of datasets from each set of word embeddings: one considering three relationships and the other considering two relationships.

Finally, we selected the following set of strategies for dealing with data imbalance: focal loss function, weighted loss function, random under-sampling, Tomek links under-sampling, and over-sampling with three SMOTE variations. We evaluated the impact of the selected techniques on the performance of a standard neural network trained for classifying lexical relationships. The results showed that even though we were able to slightly surpass the baseline model in some cases, in general, the selected techniques were not able to increase its performance.

For the experiments based on the complete dataset with three relationships, the majority of the techniques demonstrated worse performance in terms of macro F1 score than the baseline. Some techniques, such as random under-sampling demonstrated an increase in other metrics. In that case, there was an enhancement in the recall for the holonymy class, but they exhibited a decline for other classes. Notably, the over-sampling methods, specifically SMOTE and Borderline SMOTE, showed a slight superiority for the FastText dataset in comparison to the baseline results. They were able to enhance the recall for the holonymy class by approximately 0.05, with minor trade-offs, showing their potential in specific use-cases where the recall of the holonymy classes is of greater importance.

The results from the experiments with two relationships were similar to those obtained with the complete dataset, but boasted improved F1 scores. That increase indicates that relations that cause many misclassifications can result in substantial gains in terms of F1 scores if removed. The differences between recall and precision were notable in these experiments, with the baseline model having better precision than recall. But, further experiments on loss function changes and under-sampling methods highlighted the challenges of improving one metric without compromising the other. The results of the over-sampling experiments with SMOTE echoed the baseline results, and compared to the results with three relationships, they also increased recall of the holonymy class, proving again the consistency of those results.

In summary, while certain methods, especially the SMOTE variations and Tomek links under-sampling, showed promise, they primarily echoed the performance of the baseline model, with a reasonable increase in the consumption of computing resources. Hence, based on these results, the baseline model remains the most effective solution for the present classification task.

Future works can investigate approaches for classifying semantic relationships using contextual word embeddings and pre-trained language models like BERT, as those can consider multiple meanings of words, as opposed to aggregate those meanings into a single embedding with static word embeddings, as that may have affected the performance of the classification task. It is also important to investigate other neural network architectures to find a better balance between recall and precision across various classes. As the data imbalance techniques failed to provide substantial improvement, future works could also investigate specific techniques for mitigating data imbalance designed to deal with the particularities of this task. Other hybrid approaches combining under-sampling and over-sampling techniques could also be combined, as we only tested one such approach with SMOTE combined with Tomek links. Another avenue that could be explored is not excluding the pairs of related words that were related by more than one relationship and dealing with a multi-label classification problem.

REFERENCES

BATISTA, G.; BAZZAN, A.; MONARD, M.-C. Balancing training data for automated annotation of keywords: a case study. In: . [S.l.: s.n.], 2003. p. 10–18.

BOJANOWSKI, P. et al. **Enriching Word Vectors with Subword Information**. 2017.

BOJANOWSKI, P. et al. **Enriching Word Vectors with Subword Information**. 2017.

BORST, W. N. Construction of engineering ontologies for knowledge sharing and reuse. 1999.

CHEN, L. et al. A deep learning based method for extracting semantic information from patent documents. **Scientometrics**, v. 125, p. 289–312, 07 2020.

DASTRES, R.; SOORI, M. Artificial neural network systems. **International Journal of Imaging and Robotics**, v. 21, p. 13–25, 03 2021.

DEVLIN, J. et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Available from Internet: <https://aclanthology.org/N19-1423>.

ELKAN, C. The foundations of cost-sensitive learning. **Proceedings of the Seventeenth International Conference on Artificial Intelligence: 4-10 August 2001; Seattle**, v. 1, 05 2001.

FACELI, K. et al. **Inteligência artificial: uma abordagem de aprendizado de máquina**. [S.l.]: LTC, 2011.

GASMI, H.; LAVAL, J.; BOURAS, A. Cold-start cybersecurity ontology population using information extraction with lstm. In: **2019 International Conference on Cyber Security for Emerging Technologies (CSET)**. [S.l.: s.n.], 2019. p. 1–6.

GOODFELLOW, I. J.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016.

HAMMING, R. W. Error detecting and error correcting codes. **The Bell System Technical Journal**, v. 29, n. 2, p. 147–160, 1950.

HAN, H.; WANG, W.-Y.; MAO, B.-H. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In: HUANG, D.-S.; ZHANG, X.-P.; HUANG, G.-B. (Ed.). **Advances in Intelligent Computing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 878–887. ISBN 978-3-540-31902-3.

HE, H.; GARCIA, E. A. Learning from imbalanced data. **IEEE Transactions on Knowledge and Data Engineering**, v. 21, n. 9, p. 1263–1284, 2009.

HOGAN, A. et al. Knowledge graphs. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 54, n. 4, jul 2021. ISSN 0360-0300. Available from Internet: <https://doi.org/10.1145/3447772>.

HOSSEINI, M. B. et al. Ambiguity and generality in natural language privacy policies. In: **2021 IEEE 29th International Requirements Engineering Conference (RE)**. [S.l.: s.n.], 2021. p. 70–81.

JEPPESEN, J. et al. A cloud detection algorithm for satellite imagery based on deep learning. **Remote Sensing of Environment**, v. 229, p. 247–259, 08 2019.

JI, S. et al. A survey on knowledge graphs: Representation, acquisition, and applications. **IEEE Transactions on Neural Networks and Learning Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 33, n. 2, p. 494–514, feb 2022. Available from Internet: <https://doi.org/10.1109%2Ftnnls.2021.3070843>.

JURAFSKY, D.; MARTIN, J. **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**. 3. ed. [S.l.]: Pearson Prentice Hall, 2023.

KHADIR, A. C.; GUESSOUM, A.; ALIANE, H. Ontological relation classification using wordnet, word embeddings and deep neural networks. In: CHIKHI, S. et al. (Ed.). **Modelling and Implementation of Complex Systems**. Cham: Springer International Publishing, 2021. p. 136–148. ISBN 978-3-030-58861-8.

KONOPKA, B. M. Biomedical ontologies—a review. **Biocybernetics and Biomedical Engineering**, v. 35, n. 2, p. 75–86, 2015. ISSN 0208-5216. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0208521614000503>.

LEEVY, J. et al. A survey on addressing high-class imbalance in big data. **Journal of Big Data**, v. 5, 11 2018.

LEZAMA-SáNCHEZ, A. L.; VIDAL, M. T.; REYES-ORTIZ, J. A. An approach based on semantic relationship embeddings for text classification. **Mathematics**, MDPI AG, v. 10, n. 21, p. 4161, Nov 2022. ISSN 2227-7390. Available from Internet: <http://dx.doi.org/10.3390/math10214161>.

LIN, T. et al. Focal loss for dense object detection. **CoRR**, abs/1708.02002, 2017. Available from Internet: <http://arxiv.org/abs/1708.02002>.

LIU, Q.; KUSNER, M. J.; BLUNSOM, P. **A Survey on Contextual Embeddings**. 2020.

MIKOLOV, T. et al. **Efficient Estimation of Word Representations in Vector Space**. 2013.

MILLER, G. A. Wordnet: A lexical database for english. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 38, n. 11, p. 39–41, nov 1995. ISSN 0001-0782. Available from Internet: <https://doi.org/10.1145/219717.219748>.

MILLER, G. A. et al. Introduction to WordNet: an on-line lexical database. **International Journal of Lexicography**, v. 3, n. 4, p. 235–244, 1990. Available from Internet: <http://wordnetcode.princeton.edu/5papers.pdf>.

OUSSAID, M.; BOUARAB-DAHMANI, F.; CULLOT, N. Food ontology enrichment using word embeddings and machine learning technologies. In: **2022 5th International Symposium on Informatics and its Applications (ISIA)**. [S.l.: s.n.], 2022. p. 1–6.

PETERS, M. E. et al. **Deep contextualized word representations**. 2018.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: a modern approach**. 4. ed. [S.l.]: Pearson, 2020.

SUN, Q. et al. Validating auto-suggested changes for snomed ct in non-lattice subgraphs using relational machine learning. **Studies in health technology and informatics**, v. 264, p. 378–382, 08 2019.

TOMEK, I. Two modifications of cnn. **IEEE Transactions on Systems, Man, and Cybernetics**, SMC-6, n. 11, p. 769–772, 1976.

Wikipedia contributors. **Wikipedia, The Free Encyclopedia**. 2004. [Online; accessed 27-July-2023]. Available from Internet: <https://en.wikipedia.org>.