

Interpretable Deep Learning Methods for Classifying Folktales According to the Aarne-Thompson-Uther Scheme

Duarte Pinto Pompeu

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Prof. David Martins de Matos

Examination Committee

Chairperson: Prof. Alexandre Paulo Lourenço Francisco
Supervisor: Prof. Bruno Emanuel Da Graça Martins
Member of the Committee: Prof^a. Maria Luísa Torres Ribeiro Marques da Silva Coheur

May 2019

Acknowledgments

Firstly, I would like to thank my M.Sc. supervisors, Prof. Bruno Martins and Prof. David Matos, who were constantly available to discuss ideas, offer insights and provide valuable feedback on preliminary versions of this document.

I would also like to thank all my friends and colleagues with whom I have shared this academic journey, especially Daniel Gonçalves, Daniel Mendes, José Raposo, Leonardo Marques and Márcio Santos, for their advice and companionship.

Last but not least, I would like to express all my gratitude to my parents, Alcina Pompeu and Fernando Pompeu, who were always there for me and supported me in so many ways.

Resumo

Contos e lendas populares são obras de interesse histórico e literário, dado que refletem a cultura das comunidade de onde surgiram e que os preservou. Já existem alguns trabalhos anteriores em relação à análise automática deste tipo de textos, principalmente através de modelos lineares e representações *bag-of-words* para tarefas de classificação deste tipo de textos. No entanto, trabalhos mais recentes têm mostrado que arquiteturas neuronais obtêm um desempenho comparativamente superior noutras tarefas de processamento de língua natural e classificação de texto. Como tal, a minha proposta é avaliar o uso de uma arquitetura de redes neuronais profundas baseada na Rede de Atenção Hierárquica (HAN) para classificar textos multilíngues correspondentes a contos e lendas populares, tal como para gerar visualizações que facilitem a interpretação de resultados. Através de experiências, é demonstrado um desempenho superior ao de modelos lineares para classificar os textos de acordo com as categorias Aarne-Thompson-Uther. É também demonstrada a utilidade de atenção neuronal como método para gerar visualizações intuitivas dos resultados.

Palavras-chave: Análise Computacional de Contos e Lendas Populares, Classificação de Texto, Redes Neuronais Profundas, Aprendizagem Inter-Linguística

Abstract

Folktales are works of historic and literary importance, reflecting the culture of those who created and preserved them. There has been limited research regarding the automated analysis of this type of text, using mostly linear models and bag-of-words representations for automatic classification. However, recent research shows that neural network models comparatively achieve a superior performance for similar NLP tasks. As such, I propose to evaluate the use of a cross-language deep neural network approach based on the previously proposed Hierarchical Attention Network to classify multi-lingual folktales, as well as to generate visualizations to aid explaining the predictions. Results show an increased performance over baselines when predicting Aarne-Thompson-Uther categories, and I demonstrate the usefulness of neural attention as a method for generating intuitive visualizations of results.

Keywords: Computational Folkloristics, Text Classification, Deep Neural Networks, Cross-Language Learning

Contents

Acknowledgments	iii
Resumo	v
Abstract	vii
List of Tables	xi
List of Figures	xiii
Glossary	1
1 Introduction	1
1.1 Introduction	1
1.2 Thesis Proposal	2
1.3 Results and Contributions	3
1.4 Organization of the Dissertation	3
2 Concepts and Related Work	5
2.1 Representing Textual Information	5
2.1.1 Space Vector Model	5
2.1.2 Word Embeddings	7
2.2 Introduction to Neural Networks	7
2.2.1 Single and Multi-Layer Perceptrons	8
2.2.2 Recurrent Neural Networks	10
2.2.3 Convolutional Neural Networks	12
2.3 Categorization and Classification of Folktales	13
2.3.1 The Aarne–Thompson-Uther Folktale Classification System	14
2.3.2 Automatic Classification of Folktales	14
2.3.3 Folktale Classification Using Learning to Rank	15
2.4 Deep Learning Methods for Text Classification	15
2.4.1 Hierarchical Attention Network for Document Classification	15
2.4.2 Deep Pyramid Convolutional Networks for Text Labeling	18
2.4.3 Interpretable Neural Networks for Text Categorization	19
2.5 Summary	21

3	The Deep Neural Model for Classification of Folk Tales	23
3.1	The Neural Architecture	23
3.2	Cross Language Word Embeddings	25
3.3	The KNN Augmentation	25
3.4	Summary	27
4	Experimental Evaluation	29
4.1	The MFTD Dataset	29
4.2	Experimental Methodology	29
4.3	Experimental results	30
4.4	Interpreting Results by Visualizing Attention Weights	31
4.5	Summary	34
5	Conclusions and Future Work	37
5.1	Summary of Contributions	37
5.2	Future Work	38
	Bibliography	39
A	Word clouds for categories	43

List of Tables

1.1	The categories and subcategories from the Aarne-Thompson-Uther (ATU) hierarchy. . . .	2
4.1	Statistics for the MFTD dataset concerning labeled examples.	30
4.2	Evaluation results for the MFTD examples in English.	31
4.3	Evaluation results for the complete set of MFTD examples, in English, Portuguese, Spanish, French, and Italian.	31
4.4	F ₁ scores for each category.	31
4.5	F ₁ scores concerning categories for each language.	32
4.6	Ablation experiments for the HAN with multi-lingual examples from the MFTD dataset. . .	32
4.7	The 20 words with most total attention per category.	33

List of Figures

2.1	Two dimension projecting of word2vec embeddings, as shown in tensorflow.org ¹	6
2.2	A feed-forward neural network with two hidden layers, as exemplified by Goldberg [7].	8
2.3	A convolution applied over a sentence, as depicted by Goldberg [7].	13
2.4	Visualization of classified examples from Yelp 2013, as shown in Yang et al. [10].	18
2.5	Comparison between DPCNN (left) and ResNet (right), a previous CNN model, adapted from Johnson and Zhang [19]. DPCNN maintains the number of feature maps, while for ResNet they increase exponentially.	19
2.6	Highlighting texts according to Layerwise Relevance Propagation (LRP) values for input words, as displayed by Arras et al. [21].	20
3.1	The KNN augmented Hierarchical Attention Network.	24
3.2	2-dimensional projection of word embeddings in English and (aligned) Spanish.	26
4.1	Web page generated for the tale <i>The Godfather</i> , originally written by Jacob and Wilhelm Grimm. ²	33
4.2	Highlighted excerpts for the English and French versions of the <i>Cat and Mouse Partnership</i> tale, originally written by Jacob and Wilhelm Grimm. ³	34
4.3	Left: word cloud for the English version of the <i>Cat and Mouse Partnership</i> tale. Right: word cloud for English <i>Animal tales</i>	34
A.1	Word clouds for the category Animal tales.	43
A.2	Word clouds for the category Tales of magic.	44
A.3	Word clouds for the category Religious tales.	44
A.4	Word clouds for the category Realistic tales.	45
A.5	Word clouds for the category Tales of the stupid ogre.	45
A.6	Word clouds for the category Anecdotes and jokes.	46
A.7	Word clouds for the category Formula tales.	46

Chapter 1

Introduction

1.1 Introduction

Folktales are stories created by individuals or entire communities, which are traditionally passed down from generation to generation, mainly by telling. They can appear in the form of tales (e.g., fairy tales, tall tales, etc.), proverbs, myths, or jokes, and they can be used for teaching, disciplining children, or for entertainment. In contrast to high culture, where a work belongs to a named artist or organization, these works are mostly developed within a community, which nurtures, modifies, and shares them. This gives folktales an historic importance, as they reflect the culture of those who created and preserved them. As such, folktales have been important objects of study by academics in the humanities, e.g. within the context of literary and historic research.

The formal study of this domain is referred to as folkloristics. To study the culture and beliefs of a given society or community, this research discipline relies on a general process with three parts: collection of tales, archival and publishing, and analysis. Advancements in computing technology have aided this process, giving origin to the specialized sub-discipline of computational folkloristics, which concerns the use of technology for the benefit of folkloristics, e.g. managing digital collections of folktales, or developing software to improve analysis [1].

Many folk tales have been collected and archived in digital collections over the last years, such as the Archive of Portuguese Legends (APL)¹, the Multilingual Folk Tale Database (MFTD)², the Dutch Folktale Database [2], the Estonian Folk Archives [3], and several other examples [4]. These collections include a variety of texts, written in multiple languages and organized according to different classifications.

To organize and classify folktales, distinct indexes are commonly employed. One notable example is the Aarne-Thompson-Uther (ATU) classification system, composed by 2400 different index nodes, and organized in a complex hierarchy of genres, sub-genres, and so on. The first two levels of the hierarchy, which will be referred to as categories and sub-categories, are displayed in Table 1.1.

In connection to Computational Folkloristics, there has been research regarding automatic classification of folktales. For instance, machine learning models have been used to classify works from the

¹www.lendarium.org

²www.mftd.org/

Table 1.1: The categories and subcategories from the ATU hierarchy.

Categories	Subcategories
Animal tales	Wild animals, Wild and domestic animals, Animals and humans, Domestic animals, Others
Tales of magic	Adversaries, Enchanted relative, Supernatural Tasks, Supernatural Helpers, Magic Objects, Power or Knowledge, Others
Religious tales	God, Truth, Heaven, Devil, Others
Realistic tales	Marrying the princess, Marrying the prince, Fidelity and innocence, The obstinate wife, Good precepts, Clever acts and words, Tales of fate, Robbers and murderers, Others
The stupid ogre	Labor contract, Man and ogre, Man against ogre, Man kills ogre, Man frightens ogre, Man outwits the devil, Souls saved from devil
Anecdotes/Jokes	Fools, Married couples, Women, Men, Religion, Tall tales, Others
Formula tales	Cumulative tales, Catch tales, Others

the Dutch folktale database, namely SVMs [5], as well as Learn to Rank methods [6], showing moderate success in this task.

However, in other text classification tasks, this type of algorithms (i.e. linear models leveraging bag-of-words features) is often outperformed by other machine learning models. One such example are deep neural networks, which have been used for natural language processing (NLP) tasks such as classification and sentiment analysis [7]. One notable neural architecture is the Hierarchical Attention Network (HAN), which has been used to classify text by taking into account its structure of words and sentences. There has also been research in providing insights to these models' predictions, e.g. by highlighting words which internally produced higher attention weights for a given classification [8, 9, 10].

1.2 Thesis Proposal

In the context of this dissertation, I propose a deep learning model for classifying folktales, as well as for explaining its predictions. This approach involves:

- Adapting different components from state-of-the-art deep learning architectures (e.g. ideas such as the HAN, the penalized hyperbolic tangent as an activation function, the use of sparsemax attention, and a KNN augmentation) to design a model for the classification of multilingual texts.
- Implementing, training and testing this model, specifically by using folktales from the MFTD dataset, and cross validating results using a stratified k-fold approach, measuring predictions in terms of micro and macro-averaged F_1 scores.
- Studying the contribution of each component and some hyper parameters to the overall performance by performing ablation studies.
- Generating and observing visualizations based on the attention weights during predictions, using them to highlight the corresponding terms and sentences.

1.3 Results and Contributions

The main results and contributions of this work are:

- A cross-language deep learning method for encoding multi-lingual documents, which modifies and extends the original HAN by employing sparsemax attention mechanisms, the penalized hyperbolic activation function, and a KNN augmentation.
- A method to generate visualizations which help explaining predictions, by highlighting words and sentences according to the attention weights attributed to their respective encodings.
- A model achieving improved performance for predicting ATU categories, compared to baselines adapted from previous research, specifically micro and macro-averaged F_1 scores of 0.712 and 0.358, compared to 0.710 and 0.282, on the English subset of the MFTD.
- As far as I am aware, the first research using deep learning for the classification and visualization of predictions in the field of folkloristics.

1.4 Organization of the Dissertation

The rest of this dissertation is organized as follows:

- Chapter 2 covers the fundamental concepts and the related work, analyzing previous relevant research.
- Chapter 3 describes the architecture of the deep learning model in detail.
- Chapter 4 details the datasets, the experimental methodology and results.
- Chapter 5 provides a conclusion to this endeavor, including some suggestions for future work.

Chapter 2

Concepts and Related Work

Before conducting this research, it is important to study its underlying fundamental concepts, and to review work relating to the domains of folk tales and deep learning. This chapter is organized as follows:

- Section 2.1 concerns the representation of text in machine learning.
- Section 2.2 studies some fundamental concepts regarding neural networks for text classification.
- Section 2.3 reviews related work concerning automatic folktale classifiers.
- Section 2.4 regards state of the art models for text classification.
- Section 2.5 provides a summary of this chapter.

2.1 Representing Textual Information

In computer science, words and documents are commonly represented as strings, i.e. series of characters which are implemented as arrays of bytes using a certain character encoding. However, these text contents are generally converted into different data types or models that more meaningful to NLP algorithms, and which may also be more efficient to process.

2.1.1 Space Vector Model

One example of an alternative representation of text is the vector space model, i.e. an algebraic model that represents documents as vectors. In this model, each document from a given collection is represented as a sparse vector D , with dimensionality corresponding to the collection's vocabulary V . For each index, D_i represents the occurrence of the term V_i in the document.

One method of expressing this information are binary weights, which simply indicate whether a word occurs in a document or not. This is mathematically expressed as follows:

$$D_i = \begin{cases} 1 & \text{if } V_i \in D \\ 0 & \text{if } V_i \notin D \end{cases} \quad (2.1)$$

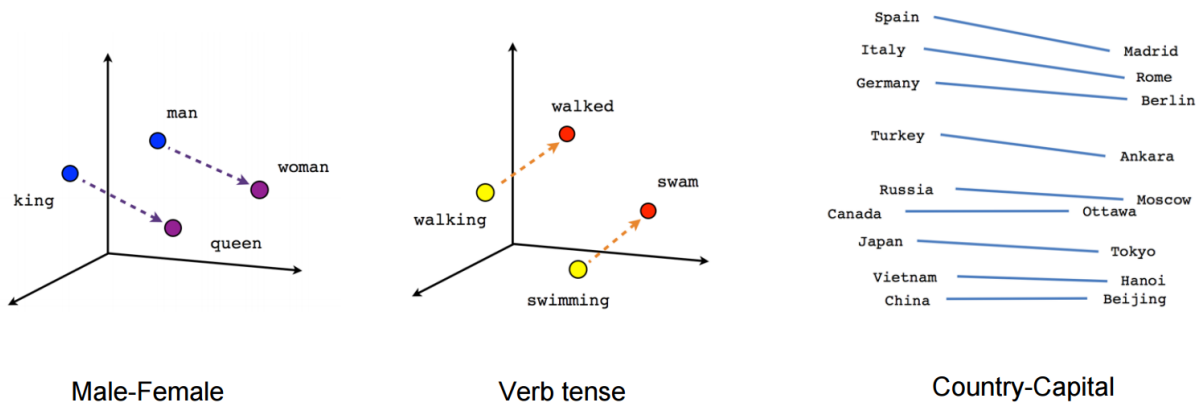


Figure 2.1: Two dimension projecting of word2vec embeddings, as shown in tensorflow.org¹.

Other weighting methods, which may be more informative, are also commonly used. Term frequency indicates how often a word occurs in a document, attributing higher weights to terms that appear more often in a given text. Inverse document frequency expresses the rarity of a word across different documents, attributing higher weights to terms that appear in fewer items. This diminishes the weight of generally common words, as well as terms that may be frequent in a given domain. For example, words such as *a* or *the* are very common in any English document. On the other hand, the distinction of other words may vary according to context, e.g. the word *astronaut* might be a considerable differentiator for documents from a general newspaper but not as much for a collection of NASA reports.

The *tf-idf* heuristic combines both these ideas. The following equations represents one approach to calculate it, where t and d represent terms and documents, while D is the set of all documents:

$$\text{tf-idf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \times \log \frac{|D|}{|\{d \in D : t \in d\}|} \quad (2.2)$$

After converting documents to sparsely weighted vectors, their content can be analyzed through algebra operations. For instance, one prominent task in information retrieval is finding relevant documents to a given query, and possibly ranking them according to a relevance score. This metric can be computed using different mathematical functions, such as vector products or cosine similarity:

$$\text{vector product}(D, Q) = \sum_{i=1}^V D_i \times Q_i \quad (2.3)$$

$$\text{cosine similarity}(D, Q) = \frac{\sum_{i=1}^V D_i \times Q_i}{|D| \times |Q|} \quad (2.4)$$

These features can also be used for document classification. For example, a *tf-idf* representation can be used for classification with support vector machines (SVMs), which by performs separations of hyper-

¹tensorflow.org/tutorials/word2vec

planes, where each plane corresponds to a category. A k -nearest neighbors (KNN) model can also use these features to find similar texts, and predict a classification based on the label from neighbors.

2.1.2 Word Embeddings

A different technique to represent text in NLP is the embedding of words i.e. each term is defined by a dense vector of values which loosely represents its semantic meaning, as may be seen in Figure 2.1. As such, documents are converted into a collection of embeddings, where each word is replaced by the an array of values.

These embeddings are often obtained from unsupervised training methods, i.e. learned from a set of unlabeled examples. This is possible by employing techniques from the field of distributional semantics, which studies similarities between words according to their distribution over a text. This is possible as a result of a property derived from the distributional hypothesis, which informally states that *linguistic items with similar distributions have similar meanings*.

In this field, there are two distinct approaches to study the distribution and context of words in a text. One is the family of *count based* methods, which analyze the statistics of co-occurrence between words and their neighbors, mapping this information into small dense vectors for each term [11]. The other is the family of *predictive* methods, which perform a similar task but models such as neural networks [12].

In terms of predictive methods, neural networks can learn the embedding vectors according to different models. One of them is the Continuous Bag of Words (CBOW) architecture associated to the word2vec tool, which aims to predict target words based on surrounding words, and can be described as *predicting the word according to its context* [13]. This model aims to maximize the log probability for word w_t and context h :

$$\log P(w_t|h) \tag{2.5}$$

On the other hand, the Skip-gram architecture takes a contrasting approach, predicting surrounding words based on target words, which can be described as *predicting the context according to a word* [13]. This model aims to maximize the following average log probability for training words w_1, w_2, \dots, w_T :

$$\frac{1}{T} \sum_T^{T=1} \sum_{-c < j < c} \log p(w_{t+j}|w_t) \tag{2.6}$$

The embedding of words is a prevalent method to translate text into features for neural network models, which are described in the following section.

2.2 Introduction to Neural Networks

Neural networks are machine learning models that have been successful in fields such as image recognition, speech processing, and NLP. They use supervised learning, i.e. are trained with labeled data, in order to learn how to perform tasks such as classification, regression, and natural language generation.

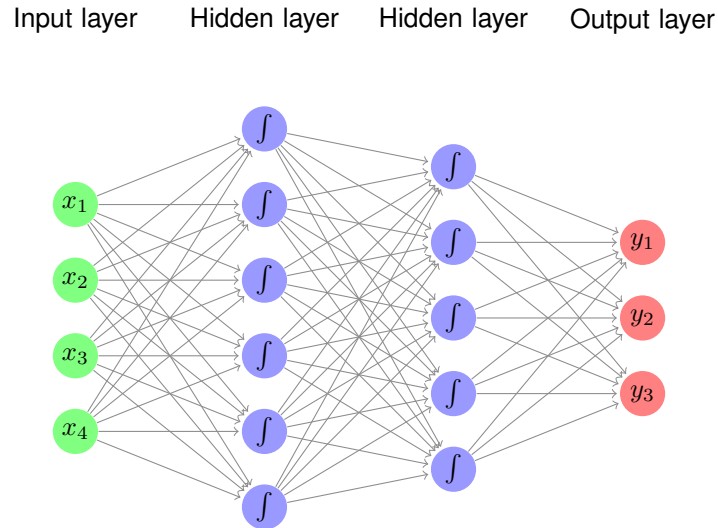


Figure 2.2: A feed-forward neural network with two hidden layers, as exemplified by Goldberg [7].

2.2.1 Single and Multi-Layer Perceptrons

Their design is inspired by a model of the human brain: in a likeness to neurons and axioms, there are computation units and connections amidst them, composing a neural network. When performing a task, the input layer introduces information to the neural network, which will be combined and processed by in a hidden layer of nodes, which then transmit the processed information to the output layer, which predict an answer. In deep learning models, the neural network is composed of multiple hidden layers, where information is combined and processed by each consecutive layer, until transmitting it to the next hidden or output layer, as is exemplified in Figure 2.2.

Mathematically, this process consists in a sequence of vector-matrix operations [7]. For instance, the simplest neural network, the perceptron, is a linear function of its inputs, with:

$$\text{NN}_{\text{perceptron}}(x) = xW + b \quad (2.7)$$

In the previous equation, W is the weight matrix, x is the input vector, and b is the bias term. The bias value is important to improve results, as it enables shifting values by means of addition. For more complex computations, hidden layers may be introduced, forming a multi layer perceptron (MLP):

$$\text{NN}_{\text{MLP}}(x) = g(xW^1 + b^1)W^2 + b^2 \quad (2.8)$$

In the previous expression, x is the input vector, W^1 and W^2 are the weights matrices, b^1 and b^2 are the bias terms, and g is the activation function. g is often a non-linearity, such as the sigmoid ($\frac{1}{1+e^{-x}}$), the hyperbolic tangent ($\frac{e^{2x}-1}{e^{2x}+1}$), or the Rectified Linear Unit (ReLU) ($\max[0, x]$) function.

In neural networks, obtaining the optimal values for the learning parameters, i.e. weights, embeddings and biases, is achieved by training the network with an extensive labeled dataset. To accomplish this, training algorithms aim to minimize a loss function, by repeatedly processing input from the training

Algorithm 1 The Adam optimizer, as seen in Goodfellow et al. [14].

Require: α Stepsize**Require:** $\beta_1, \beta_2 \in [0, 1]$ Exponential decay rates for moment estimates**Require:** $f(\theta)$ Stochastic objective function with parameters θ **Require:** θ_0 Initial parameter vector $m_0 \leftarrow 0$ (Initialize 1st moment vector) $v_0 \leftarrow 0$ (Initialize 2nd moment vector) $t \leftarrow 0$ (Initialize timestep)**while** θ_t not converged **do** $t \leftarrow t + 1$ $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t) $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate) $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate) $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate) $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate) $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)**end while****return:** θ_t (Resulting Parameters).

set, estimating the error (the difference between the prediction and expected result), and adjusting the learning parameters according to the gradient's direction. One of these optimization algorithms is the online stochastic gradient descent (SGD) procedure, which iterates over the training set, updating the learning parameters as follows:

$$\theta = \eta \Delta(L(f(x, \theta), y)) \quad (2.9)$$

For each cycle, the learning parameters θ are updated in the gradient's Δ direction. Instead of adopting the current best value, they are scaled by a learning rate η , i.e. a hyper-parameter which adjusts the speed rate of the training. This is often a low value, which decreases the likelihood of missing local minima, at the cost of increased computation time.

Besides SGD, other approaches are also commonly used, such as Adam [14]. The name Adam is derived from adaptive moment estimation, where individual learning rates are adapted according to estimates of first and second moments, as described in Algorithm 1. This attempts to combine the advantage of two other optimizers: AdaGrad, which has been successful with sparse gradients, and RMSProp, which works well with online and non-stationary settings.

These optimization algorithms require a method for calculating the gradient of parameters, which indicates how they should be adjusted for an optimal local result. One such method is the back propagation procedure: after representing the neural network as a computation graph, the forward pass computes the values of each computation node, and then a backward pass calculates the partial derivatives of each node, to obtain the gradients of each learning parameter, according to multi-variable differential calculus concepts. This indicates how they should be adjusted to reach local optimum values, which optimizers such as SGD and Adam use to minimize loss in each iteration.

Regarding the input of neural networks in NLP, deep learning models commonly use vectors of feature embeddings, i.e. learned representations of words. When the embeddings were previously learned from unsupervised training, it is useful to map them to the first layer of the neural network, which is then called the embedding layer. This enables the embeddings of words to be adjusted and optimized during training, improving the representations for the task's particular domain. In this case, an input to the neural network is commonly a succession of word indexes, whose embedding is calculated after a forward pass in the layer.

On the other hand, the output of a neural network is computed by the last layer. The structure of this layer is modeled according to the labels from the dataset, e.g. if a dataset has examples from three different categories, the output layer will commonly have three nodes. The result of this layer is also dependent on the labels seen during training, such as classifications, e.g. the node with the highest value represents the predicted category.

It is often useful to transform these values into into a probability distribution, allowing us to measure them more easily. One way to achieve this is using the softmax function, expressed as follows:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \quad (2.10)$$

This function transforms each value into a non-negative number such that the sum of all values is exactly 1. In single label problems, i.e. predictions where only one simultaneous outcome can occur, the highest value represents the prediction.

A deep neural network follows the same architecture as described for a general MLP, but specifically has more than one hidden layer between the input and output layers. This enables deep networks to learn by composing features from previous layers, in a feature hierarchy. Other ways to improve learning are achieved by using different architectures, such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs).

2.2.2 Recurrent Neural Networks

Besides feed forward models, other architectures have been proposed. One that is specifically good at solving problems regarding time-series or word sequences is the recurrent neural network (RNN). A RNN takes an ordered list of input vectors x_1, \dots, x_n and returns both an ordered list of states s_1, \dots, s_n and an ordered list of outputs y_1, \dots, y_n . Over each step i , the resulting state is obtained by applying the function R to the current input and previous state. The output for any step can be obtained by applying the function O to the corresponding state. This process is represented in the following expressions:

$$\begin{aligned} \text{RNN}(s_0, x_{1:n}) &= s_{1:n}, y_{1:n} \\ s_i &= R(s_{i-1}, x_i) \\ y_i &= O(s_i) \end{aligned} \quad (2.11)$$

Mathematically, these steps can be represented in a recursive function. For example, the 4th step of an arbitrary long sequence x_1, \dots, x_n would be:

$$\begin{aligned}
 s_4 &= \mathbf{R}(s_3, x_4) \\
 &= \mathbf{R}(\mathbf{R}(s_2, x_3), x_4) \\
 &= \mathbf{R}(\mathbf{R}(\mathbf{R}(s_1, x_2), x_3), x_4) \\
 &= \mathbf{R}(\mathbf{R}(\mathbf{R}(\mathbf{R}(s_0, x_1), x_2), x_3), x_4)
 \end{aligned} \tag{2.12}$$

Like other neural networks, RNNs have learning parameters, shared among all states. This makes training possible by *unrolling the recursion*, i.e. creating a computation graph where each step is represented as a different network layer. This results in a structure equivalent to a deep feed forward neural network, therefore most algorithms for the previous models can be adapted for RNNs. One example is the back propagation through time, which is used to calculate gradients in recurrent models.

RNN architectures can be distinguished between different variants. The Simple-RNN (S-RNN) model is possibly the most straight forward, represented as follows:

$$\begin{aligned}
 s_i &= \mathbf{R}_{\text{SRNN}}(s_{i-1}, x_i) = g(x_i W^x + s_{i-1} W^s + b) \\
 y_i &= \mathbf{O}_{\text{SRNN}}(s_i) = s_i
 \end{aligned} \tag{2.13}$$

In the previous expressions, \mathbf{R}_{SRNN} is a non-linear activation function g applied to the current input and previous state, while \mathbf{O}_{SRNN} simply returns the input state, i.e. for any given step i , $y = s_i$.

However, training this RNN variant may be troublesome, due to the vanishing gradients problems. This is a predicament caused by the growth of input sequences, where gradients are diminished in the later steps, and error is propagated with less effectiveness for the earlier inputs. The Long Short-Term Memory (LSTM) architecture aims to solve this problem by introducing a memory component, which improves the preservation of gradients over time [15]. Access to this memory is controlled with gates which limit how much information is added or forgotten in the memory cells. These gates are specifically a $[0 : 1]^n$ vector, where values are skewed towards 0 or 1, e.g. by using a sigmoid function. In these gates, an index with value close to 1 is allowed to pass, while a value close to 0 is blocked. This architecture can be described by the following expressions:

$$\begin{aligned}
s_j &= \text{RLSTM}(s_{j-1}, x_j) = [c_j; h_j] \\
c_j &= c_{j-1} \odot f + g \odot i \\
h_j &= \tanh(c_j) \odot o \\
i &= \sigma(x_j W^{xi} + h_{j-1} W^{hi}) \\
f &= \sigma(x_j W^{xf} + h_{j-1} W^{hf}) \\
o &= \sigma(x_j W^{xg} + h_{j-1} W^{hg}) \\
g &= \tanh \\
y_j &= \text{OLSTM}(s_j) = h_j
\end{aligned} \tag{2.14}$$

In the previous expressions, \odot represents the component-wise product operation, c_j is the memory component and h_j is the output at time j . The input, forget, and output gates are i , f , and o , correspondingly. The update of the memory component is controlled by i and f , which define how much is learned from the input and how much is forgotten, respectively. The current state and output is obtained from the memory and the o gate. This gating mechanism reduces the vanishing gradients problem, with memory related gradients remaining high even after long sequences of input.

However, this architecture has one drawback: its complexity, which is inconvenient for both analysis and computation. The Gated Recurrent Unit (GRU) architecture was proposed to address this problem, consisting of a simpler model [16], and yet performing comparably. Similarly to LSTMs, it uses a gating mechanism, although it uses fewer gates and omits the dedicated memory component. This is defined by the following expressions:

$$\begin{aligned}
s_j &= \text{RGRU}(s_{j-1}, x_j) = (1 - z)s_{j-1} + z \odot h \\
z &= \sigma(x_j W^{xz} + h_{j-1} W^{hz}) \\
r &= \sigma(x_j W^{xr} + h_{j-1} W^{hr}) \\
h &= \tanh(x_j W^{xh} + (h_{j-1} \odot r) W^{hg}) \\
y_j &= \text{OLSTM}(s_j) = s_j
\end{aligned} \tag{2.15}$$

In the previous expressions, the reset gate r is used to access the previous state and computing a proposed update to h , and the update gate z is used to compute the state s_j , from an interpolation of s_{j-1} and h .

2.2.3 Convolutional Neural Networks

A different approach to make predictions from word sequences corresponds to using convolutional neural networks (CNNs), i.e. neural networks with one or more convolution layers. This architecture aims to

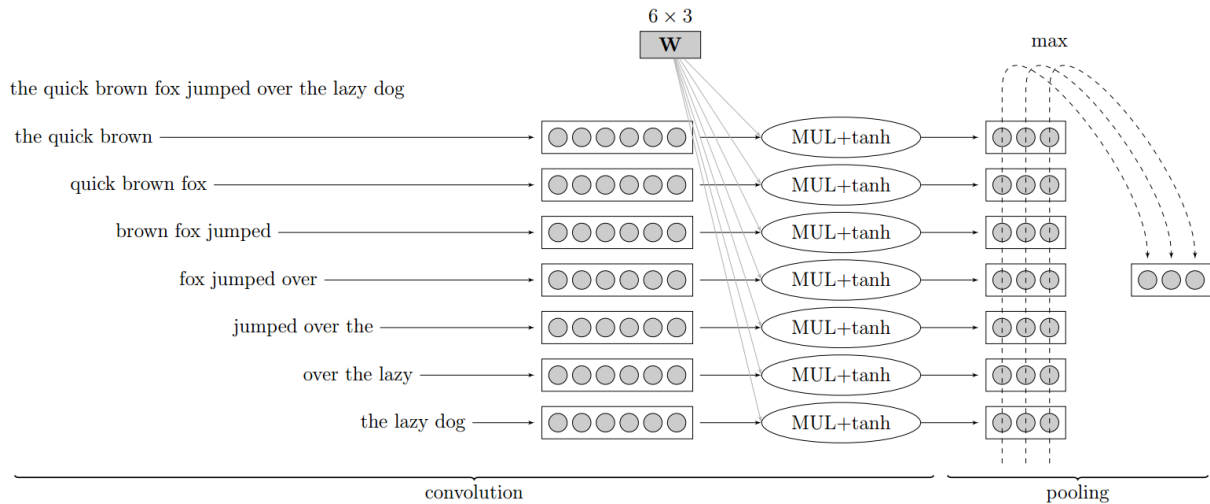


Figure 2.3: A convolution applied over a sentence, as depicted by Goldberg [7].

extract the most significant features from a large structure of ordered items, combining groups of them to form a fixed size vector representation [17].

Mathematically, a convolution is an operation that uses integral calculus to combine and merge two functions into a new one. In CNNs for text processing, a convolution operation is applied over all instances of a k -word sliding window over a sentence, using a learned filter function. Then a pooling operation combines the previous windows into a single d -dimensional vector, using either the max or average value observed. An example is displayed in Figure 2.3.

Specifically, the convolution layer takes the embeddings of a word $v(x_i)$ over a k -window sequence, and then applies a filter function over the concatenated i th windows w_i . This results in m vectors p_1, \dots, p_m such that

$$p_i = g(w_i W + b) \quad (2.16)$$

where g is a non-linear activation function, w_i the concatenated vector of the i th window, W the weights matrix and b the network bias. The filter function is a learning parameter of the neural network, thus it is adjusted during the training process according to the gradients of the network's loss function.

A following layer applies a pooling operation over p_1, \dots, p_m to obtain a single d_{conv} dimensional vector c . For example, using a *max pooling layer*, this would result in the following representation:

$$c_j = \max p_i[j] \quad (2.17)$$

2.3 Categorization and Classification of Folktales

So far, there has been minimal research in automatic classification of folktales using machine learning methods. Although neural networks were not used, it is useful to study these experimentations and gain insights which may results in improving the model to be developed for this MSc. thesis.

2.3.1 The Aarne–Thompson-Uther Folktale Classification System

In order to study a vast collection of folk tales, researchers have developed standard, hierarchical indexes to organize these works, facilitating their access and analysis. For instance, the Aarne–Thompson classification systems categorize narrations according to their type or motif. This family of systems is composed of different structures, namely the Aarne-Thompson motif index, the Aarne-Thompson tale type index, and the Aarne-Thompson-Uther classification system, which is the latest update and expansion to the previous hierarchies.

Specifically, the ATU system catalogs folktales according to their type, attributing them a number from the correspondent index. This system defines 2400 different types, which are grouped according to similarity or themes, in a complex hierarchy. Occasionally, some ATU indexes are sometimes specialized into subtypes, which are represented by the parent's index number and an additional letter. For example, the famous story *the ant and the cicada* is attributed the sub-type *ATU280A: The ants and the grasshopper*, which is a specialization of the type - *ATU280: The ant carries a load as large as itself*². Ranges of indexes are also used to form more complex hierarchies, e.g. *1-69: The Clever Fox* is a subset of *1-99: Wild Animals*, which groups with other ranges to form *1-299: Animal Tales*. The full index and its hierarchy can be seen in the MFTD website³.

2.3.2 Automatic Classification of Folktales

For the first research, a classifier for folktales was developed by Nguyen et al. [5]. It concerned works from the Dutch Folktale Database according to 9 narrative genres (distinct from the ATU categories). The dataset consisted of 8311 examples, which were multi-class and multi-label, i.e. each one could have one or more associated labels.

For this folktale classifier, the authors used a support vector machine (SVM) model. This is a supervised machine learning algorithm which represents features as points in space, and performs classification by separating these into different hyper-planes, according to their labels. This division is performed with a function, which is learned by supervised training. In this model, the authors used a linear kernel and L2 regularization, and the method by Crammer and Singer for multi-class classification [18].

The features used for representing the documents were composed of (a) lexical features, such as unigrams and character n-grams, (b) stylistic and structural features, such as part of speech (POS) unigrams, POS bigrams, punctuation, whitespace, and text statistics, (c) domain knowledge, such as automatically tagged persons, locations and events, and (d) annotated metadata, such as keywords, named entities, summary and date. The labels were the following narrative genres: *Fairy Tales*, *Legends*, *Saint's Legends*, *Urban Legends*, *Personal Narratives*, *Riddles*, *Situation Puzzles*, *Jokes* and *Songs*.

Through experimentation, the authors reported an average F_1 score of 0.62, with some misclassification of similar categories, such as fairy tales and legends, and poorer results for categories with fewer

²www.mftd.org/index.php?action=story&id=9018

³www.mftd.org/index.php?action=atu

examples. Regarding the features, characters n-grams were more effective than unigrams, and other features only provided marginal improvements to the accuracy of the model.

2.3.3 Folktale Classification Using Learning to Rank

In a following study, Nguyen et al. [6] developed a classifier which extended the SVM with a Learn to Rank mechanism. The same Dutch Folktale Database was considered, but this time using labels from the ATU index and Type-Index of Urban Legends. In this case, the final dataset was a multi-class single-label situation, i.e. there are multiple labels but each example can only have a single one.

In this research, the authors framed classifications as a ranking problem, using a Learning to Rank algorithm to predict the most-likely classifications of an example. Two baselines were considered, namely a *big document model* and a *small document model*. In the first one, documents from each category are aggregated into an individual virtual document. A prediction is performed by querying for relevant documents, with the resulting order providing a ranking of categories. In the second model, each story is a different document. In a prediction, a query will provide the most relevant documents, whose category will be considered for classification and ranking - specifically, only the top result was considered, which is similar to a k -nearest neighbor classifier with $k = 1$. This model is queried with BM25, and its rankings improved with Learning to Rank.

This model used features such as: information retrieval measures, lexical similarity, similarity to all stories of the candidate's story type, and Subject-Verb-Object (SVO) triplets. Regarding the classifications used, some examples for the ATU are *ATU 0333: Red Riding Hood* and *ATU 0275A: The Race between Hare and Tortoise*, while for the Type-Index of Urban Legends, some examples are *The Microwaved Pet*, *The Kidney Heist*, and *The Vanishing Hitchhiker*

After experimentations, the results showed a total mean reciprocal rank (MRR) accuracy of 0.82 for the ATU labels and 0.76 for the urban legends labels. The small document model was used as a baseline, as it provided the best results, which were improved by also using features from the big document model and other information retrieval queries. There was also marginal improvements using lexical and semantic features.

2.4 Deep Learning Methods for Text Classification

Although there is no known research regarding neural models for folktales classifiers, there has been comprehensive research regarding deep learning models for similar tasks of text classification. As such, it is important to study them, as they have provided state of the art models and methodologies which may be applied to this domain.

2.4.1 Hierarchical Attention Network for Document Classification

As described in section 2.2.2, recurrent networks have been useful for tasks of processing natural language, but they have also shown difficulties in some cases. For example, updating the memory compo-

ment is a complex task, since some words may be important than others to distinguish labels, i.e. not all words are important for predictions. Proper training should handle this, but it does not always succeed.

To solve this issue, Bahdanau et al. [8] proposed the attention mechanism, which attempts to distinguish elements according to their relevancy, attributing higher weights to the most suitable ones. To do so, instead of encoding an input sequence into a single vector, it uses multiple vectors that are chosen adaptively according to a computed attention. The authors describe this as *taking the weighted sum of all the annotations as computing an expected annotation*. This attention layer enables the following ones, e.g. a decoder, to retrieve information from previous states selectively, accounting for different attention weights.

In this model, the conditional probability is described as follows:

$$p(y|y_1, \dots, y_{i-1}, x) = g(y_{y-1}, s_i, c_i) \quad (2.18)$$

In the previous equation, x represents the input, y the output, s the hidden states and c_i is the context vector for target y_i . This vector is computed as a weighted sum of annotations:

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j \quad (2.19)$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (2.20)$$

$$e_{ij} = a(s_{i-1}, h_j) \quad (2.21)$$

In the previous equations, a_{ij} is the weight of each annotation h_j , and e_{ij} is the *alignment model*, and a parametrized as a feed forward neural network, trained with the other components of the RNN. In this case, the expression for computing the context vector is the `softmax` function, which will (a) turn the attention weights into a series of values between 0 and 1, and (b) guarantee their sum is exactly 1.

Using this mechanism for the task of automatic translations, the authors were able to generate target words based on the most relevant source encodings. They were also able to scrutinize the most significant words for the generation of each target word, which is an useful technique to enable visualizing or explaining predictions.

Using this concept, Yang et al. [10] proposed an hierarchical attention network, a classification model composed of RNNs and attention mechanisms. The authors considered a basic hierarchical document structure, using a composition of one RNN and attention mechanism for encoding words, and another for sentences.

In sum, this architecture is a pipeline of two RNNs and attention mechanisms. The first network creates representations of words by encoding them, and a word attention mechanism weights them according to perceived relevancy. Then, the second RNN creates representations of entire documents by encoding their sentences, and a final sentence attention mechanism weights them according to recognized significance, finally transmitting the result to a softmax function which returns the final prediction.

The specific RNN architecture used by the authors was the GRU, with a reset gate r_t and an update gate z_t . After words are embedded to vectors, these are encoded using a bidirectional GRU, which enables the model to generate representations considering the context of both previous and following contexts. Following them, a mechanism attention is used to extract the most significant representation. This is shown in the expressions below:

$$\begin{aligned} h_{x_{it}} &= \vec{h}_{it} + \overleftarrow{h}_{i(T-t)} \\ &= \overrightarrow{\text{GRU}}(x_{it}) + \overleftarrow{\text{GRU}}(x_{i(T-t)}) \end{aligned} \quad (2.22)$$

$$u_{it} = \tanh(W_w h_{it} + b_w) \quad (2.23)$$

$$\alpha_{it} = \text{softmax}(u_{it}) \quad (2.24)$$

$$s_i = \alpha_{it} h_{it} \quad (2.25)$$

After the first component of the hierarchical network, a similar process applies to the processing of sentences. To encode sequences of word encodings, an identical bidirectional GRU is used, again followed by an attention mechanism:

$$\begin{aligned} h_i &= \vec{h}_i + \overleftarrow{h}_{(L-i)} \\ &= \overrightarrow{\text{GRU}}(s_i) + \overleftarrow{\text{GRU}}(s_{L-i}) \end{aligned} \quad (2.26)$$

$$u_{it} = \tanh(W_s h_i + b_s) \quad (2.27)$$

$$\alpha_i = \text{softmax}(u_i) \quad (2.28)$$

$$v = \alpha_i h_i \quad (2.29)$$

Finally, the document vector v is normalized with a softmax function, and the training loss is computed as the log likelihood of the correct label, where j represents a label, and d , a document:

$$p = \text{softmax}(W_c v + b_c) \quad (2.30)$$

$$L = - \sum_d \log p_{dj} \quad (2.31)$$

The authors used some familiar datasets, collected from services such as Yelp, IMDB, Yahoo Answer and Amazon for training, validating and testing their model, as well as for drawing comparisons against other state of the art models. When comparing results, the hierarchical attention network achieved the highest performance for all datasets.

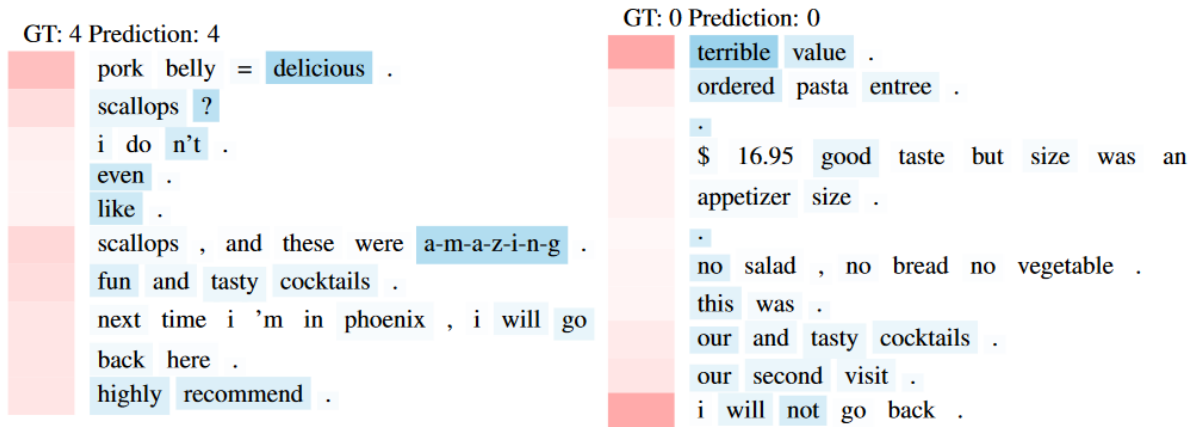


Figure 2.4: Visualization of classified examples from Yelp 2013, as shown in Yang et al. [10].

Since attention mechanisms measure a weight for each input based on its perceived importance, visualization of important inputs has been made possible by considering these values, e.g. by highlighting words in a text or pixels in a image. This visualization of relevant words and sentences is shown in some examples from the Yelp and Yahoo Answers datasets. For example, two Yelp reviews are shown in Figure 2.4, with most relevant words and sentences highlighted.

2.4.2 Deep Pyramid Convolutional Networks for Text Labeling

Another type of models which have shown to successfully classify text are the convolutional neural networks. By extending this model, Johnson and Zhang [19] proposed the Deep Pyramid CNN (DPCNN) architecture, a deeper convolutional network where computation time per layer decreases exponentially (therefore the pyramid analogy). Results show this model outperforms shallow CNNs for sentiment analysis and topic classification.

More precisely, this model is an extension of the CNN architecture, also consisting of sequences of convolution and down-sampling layers. Three aspects are considered key features of this specific architecture: down-sampling without increasing the number of feature maps, shortcutting connections with pre-activation and identity mapping, and unsupervised text region embeddings.

Down-sampling without increasing the size of feature maps is achieved by keeping the number of maps and performing a 2-stride down-sampling, which reduces the per-block computations in half. A comparison with ResNet, (a CNN which increases the number feature maps) may be observed in Figure 2.5.

Shortcut connections aim to reduce the degradation problem in deep networks, where increased depth leads to reduced accuracy, due to the difficulty in propagating errors after multiple layers. To solve this, a shortcut can be taken, where a connection can skip a certain number of layers, and thus $F(x)$ becomes $H(x) = F(x) + x$. The DPCNN architecture uses pre-activation shortcuts, where the skipped layers are two convolution layers, and the activation is performed before weighting, which the authors consider to ease the training of the neural network and improve results. All shortcuts use simple identity

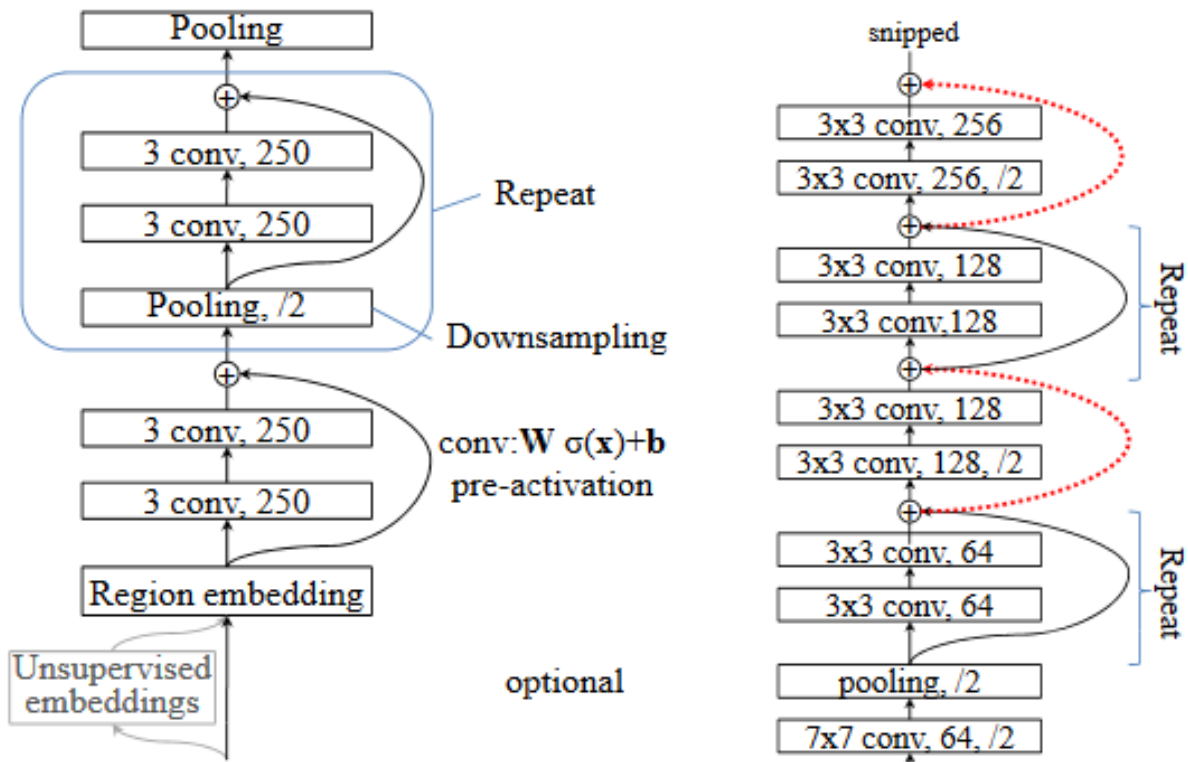


Figure 2.5: Comparison between DPCNN (left) and ResNet (right), a previous CNN model, adapted from Johnson and Zhang [19]. DPCNN maintains the number of feature maps, while for ResNet they increase exponentially.

mapping, passing data *as is*. Since shortcuts never skip down-sampling layers, x and $F(x)$ have the same dimensionality, and thus there is no need for dimension matching.

After experiments on document classification, with datasets built from services such as Yelp and Yahoo Answers, this architecture showed improved error rates when compared to ShallowCNN and other approaches. Tests with different network depths showed an error rate improvement until 15 layers, after which the error rate does not change significantly.

2.4.3 Interpretable Neural Networks for Text Categorization

It is also important to consider other aspects of machine learning models, besides the accuracy of their results. For example, another desirable feature is their ability to explain how or why they predict the outcome for a given input. Some models, such as Decision Trees or Naive Bayes, have a straight forward method of describing this process, e.g. by mapping their structure into a flow chart or computing the associated probabilities. On the other hand, it is much harder to make sense of neural networks and their thousands of learning parameters.

However, this does not mean it is impossible to obtain insights about the process. For example, Horn et al. [20] explore methods for the visualization of relevant words in a dataset. By turning words

sci.space	<p>It is the body's reaction to a strange environment. It appears to be induced partly to physical discomfort and part to mental distress. Some people are more prone to it than others, like some people are more prone to get sick on a roller coaster ride than others. The mental part is usually induced by a lack of clear indication of which way is up or down, ie: the Shuttle is normally oriented with its cargo bay pointed towards Earth, so the Earth (or ground) is "above" the head of the astronauts. About 50% of the astronauts experience some form of motion sickness, and NASA has done numerous tests in space to try to see how to keep the number of occurrences down.</p>
sci.med	<p>It is the body's reaction to a strange environment. It appears to be induced partly to physical discomfort and part to mental distress. Some people are more prone to it than others, like some people are more prone to get sick on a roller coaster ride than others. The mental part is usually induced by a lack of clear indication of which way is up or down, ie: the Shuttle is normally oriented with its cargo bay pointed towards Earth, so the Earth (or ground) is "above" the head of the astronauts. About 50% of the astronauts experience some form of motion sickness, and NASA has done numerous tests in space to try to see how to keep the number of occurrences down.</p>

Figure 2.6: Highlighting texts according to LRP values for input words, as displayed by Arras et al. [21].

from a collection of documents into a space vector representation, namely a *tf-idf* matrix, while using labeled datasets or clustering unlabeled collections, the authors explore groups of documents according to different methodologies.

When *salient features* are used, a relevancy score is computed using *tf-idf*, which successfully weighs down words present in many classes. On the other hand, the *distinctive words* approach uses traditional statistic methods to compute relevancy scores, using ratios, means and variances of word occurrence among different classes. Alternatively, using *decomposed classifier scores*, a linear SVM classifier is trained with examples from the dataset. By analyzing the input weights of its predictions, the authors obtain a word relevancy score using LRP. Then, by further inspecting all the predictions for a given category, it is possible to verify the terms which were more relevant for a given label.

The authors provide examples of their processes using a dataset of cancer research papers, showing that *man* is a relevant word for the class *prostate cancer* and *women* for *breast cancer*. In their findings, *decomposing classifiers* provided the best relevancy scores, but only if their predictions were highly accurate, which is not always achievable. As a fall back mechanism, the statistic approach to gather *distinctive words* performed well, without the need of a classification process.

These methods provide some insights about the datasets used, but not on the models' predictions themselves. However, there is also research regarding this aspect. Arras et al. [21, 22] studied the relevance of input words using Layerwise Relevance Propagation (LRP) for the task of document classification. Considering a neural network, $f(x)$ is computed with a forward pass, where the output y is obtained essentially by performing multiple combinations of weighted inputs and their activation functions, summed with a bias:

$$\begin{aligned}
z_{ij} &= x_i w_{ij} + \frac{b_j}{n} \\
z_j &= \sum_i z_{ij} \\
x_j &= g(z_j)
\end{aligned} \tag{2.32}$$

In this expression, a layer is decomposed using values from the neurons x_i , the weights w_{ij} , and the bias b_j , followed by a sum and activation into the final value x_j . Considering this, LRP performs a backward pass to decompose the function in terms of each particular feature. Then, it takes this individual weight to compute a relevancy score for each input:

$$\begin{aligned}
R_{i \leftarrow j} &= \frac{z_{ij} + \frac{z_j}{n}}{\sum z_{ij} + s(z_j)} R_j \\
R_i &= \sum_j R_{i \leftarrow j}
\end{aligned} \tag{2.33}$$

In the previous expression, $s(z_j)$ is a stabilizing term for handling division by zero or near-zero values, and R_i represents the total relevance of neuron i .

To study if this scoring mechanism accurately measures relevance, the authors performed perturbation tests with deletion of words, where an increasingly high number of words n is removed from previous correctly classified texts, leading to decreases in accuracy. By comparing this decline in results with different word deletion methods (in order of LRP score, randomly, and others), it is empirically shown that the use of LRP provides highly relevant words, whose deletion results in the steepest decrease of accuracy. The authors also carried out the opposite approach, where incorrectly classified texts were considered, and deleting the least relevant words also led to an increase in performance.

Apart from LRP, other methods can also be used to find significant words in neural models. As seen in section 2.4.1, attention mechanisms also provide weights to encoding of terms or sentences, deemed the most relevant for predictions.

In conclusion, it is possible to consider significant words to generate visualizations that explain their contribution towards a given output, which provides an interpretation for machine learning predictions. As shown in Figure 2.6, these visualizations can be based on word-clouds or highlighting of text, providing explanations to end users, while requiring little to no knowledge in machine learning.

2.5 Summary

This chapter presented a study of fundamental concepts and related work which were relevant in this dissertation. Firstly, there was a comparison of different approaches to represent text, namely the space vector model and its sparse representations, and the embedding model, which uses dense representations learned by prediction methods. These embeddings are often used as features for neural networks,

which learn how to approximate a theoretical function with supervised training. Among the different types of neural architectures, some examples are the feed forward neural network, the recurrent neural network, and the convolutional neural network. Concerning folktales, some formal approaches to formally categorize these works were studied, specifically the Aarne-Thompson-Uther (ATU) system. Related work in this field was also reviewed, namely automated classifications using linear models on a Dutch folktale dataset. Finally, there was a review of deep learning approaches for text classification, namely the DPCNN and the HAN. The last one is a hierarchical approach, using encoders of words and sentences to generate a document representation, and also employing attention layers to weight each term or sentence according to its learned relevance. Furthermore, methods to generate visualizations that explain predictions of deep learning models were also explored, namely the highlighting of words according to attention weights.

Chapter 3

The Deep Neural Model for Classification of Folk Tales

In our research, we develop a model inspired by the HAN [10, 9], but modifying the attention layer to use the sparsemax function [23] and the activation functions, which are replaced by the penalized hyperbolic tangent [24]. We also extend it with a KNN component[25], which is used to find and encode similar documents. Additionally, the proposed model considers two outputs for categories and subcategories, activated by the softmax function. To illustrate this architecture, a diagram is shown in Figure 3.1.

In the remainder of this chapter, Section 3.1 covers the neural architecture. Section 3.2 provides details about the cross-lingual embeddings used in my deep learning approach. Finally, Section 3.3 provides some details concerning the KNN augmentation and Section 3.4 provides an overview of this chapter.

3.1 The Neural Architecture

In the proposed neural architecture, both encoders are bidirectional GRUs, which are recurrent layers defined by the following equations:

$$\begin{aligned} s_j &= \text{R}_{\text{GRU}}(s_{j-1}, x_j) = (1 - z)s_{j-1} + z \odot h \\ z &= \sigma(x_j W^{xz} + h_{j-1} W^{hz}) \\ r &= \sigma(x_j W^{xr} + h_{j-1} W^{hr}) \\ h &= \text{penalized tanh}(x_j W^{xh} + (h_{j-1} \odot r) W^{hg}) \\ y_j &= \text{O}_{\text{GRU}}(s_j) = s_j \end{aligned} \tag{3.1}$$

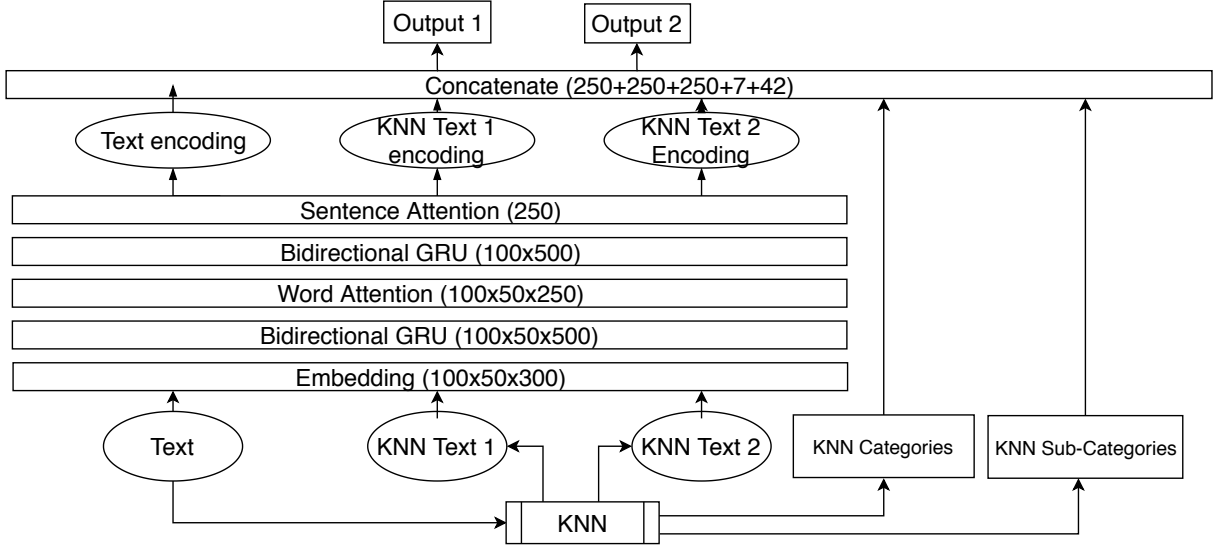


Figure 3.1: The KNN augmented Hierarchical Attention Network.

The reset gate r is used to access the previous state and computing an update to h , and the update gate z is used to compute the state s_j . To obtain further context, a bidirectional GRU is employed, concatenating sequences processed in forward and reverse order:

$$\begin{aligned}
 h_{x_{it}} &= \vec{h}_{it} + \overleftarrow{h}_{i(T-t)} \\
 &= \overrightarrow{\text{GRU}}(x_{it}) + \overleftarrow{\text{GRU}}(x_{i(T-t)})
 \end{aligned} \tag{3.2}$$

Furthermore, the penalized hyperbolic tangent function [24] was used in the activations of the GRU gates. This function is defined as follows:

$$\text{penalized tanh}(x) = \begin{cases} \tanh(x), & x > 0, \\ 0.25 \tanh(x), & x \leq 0 \end{cases} \tag{3.3}$$

Additionally, extracting and annotating the most significant terms and sentences is accomplished by attention mechanism layers. These layers employ feed forward networks leveraging the sparsemax [23] activation function, and perform weighting such that the sum of all non-negative weights is exactly 1, similarly to a probabilistic distribution. These layers are defined by the following expressions:

$$u_{it} = \text{penalized tanh}(W_w h_{it} + b_w) \tag{3.4}$$

$$\alpha_{it} = \text{sparsemax}(u_{it}) \tag{3.5}$$

$$s_i = \alpha_{it} h_{it} \tag{3.6}$$

Algorithm 2 Sparsemax evaluation

Input: z
Sort z as $z_{(1)} \geq \dots \geq z_{(K)}$
Find $k(z) := \max \left\{ k \in [K] \mid 1 + kz_{(k)} > \sum_{j \leq k} z_{(j)} \right\}$
Define $\tau(z) = \frac{(\sum_{j \leq k(z)} z_{(j)}) - 1}{k(z)}$
Output: p s.t. $p_i = [z_i - \tau(z)]_+$.

In the previous equation, u_{it} is the hidden representation resulting from a single layer MLP, a_{it} are the computed attention weights, and s_i is the new generated annotation corresponding to a single vector encoding an entire sequence.

I opted to use the sparsemax function in both attention layers, as it has been shown to improve predictions and visualizations. By generating distributions of attention weights with abundant zero values, it generates outputs with fewer, but more relevant, annotations. The procedure used for computing this function is shown in Algorithm 2.

3.2 Cross Language Word Embeddings

One challenge in our particular classification task is the multiplicity of languages present in the MFTD dataset. In order to leverage training data from different languages, pre-trained cross-lingual embeddings from the MUSE¹ project were used.

To generate these embeddings, the MUSE project took word vectors for different languages, specifically the FastText embeddings [26] trained with Wikipedia data, and then mapped them to match the English embeddings, such that a source word will hold similar values to its English translation. This aligning process was performed by employing the supervised iterative Procrustes method [27], which aims to find a transformation W that aligns embeddings X into Y :

$$W^* = \operatorname{argmin} \|WX - Y\|_F = UV^T, \text{ with } U\Sigma V^T = \operatorname{SVD}(YX^T) \quad (3.7)$$

The project also provides a demonstration for visualizing the 2-dimensional projection of word embeddings, as shown in Figure 3.2, demonstrating that aligned words are often close to the target translation.

3.3 The KNN Augmentation

Inspired by previous research [25], a KNN component was developed to find documents similar to the input text. First, the two nearest neighbors from the training dataset are discovered by employing a

¹<https://github.com/facebookresearch/MUSE>

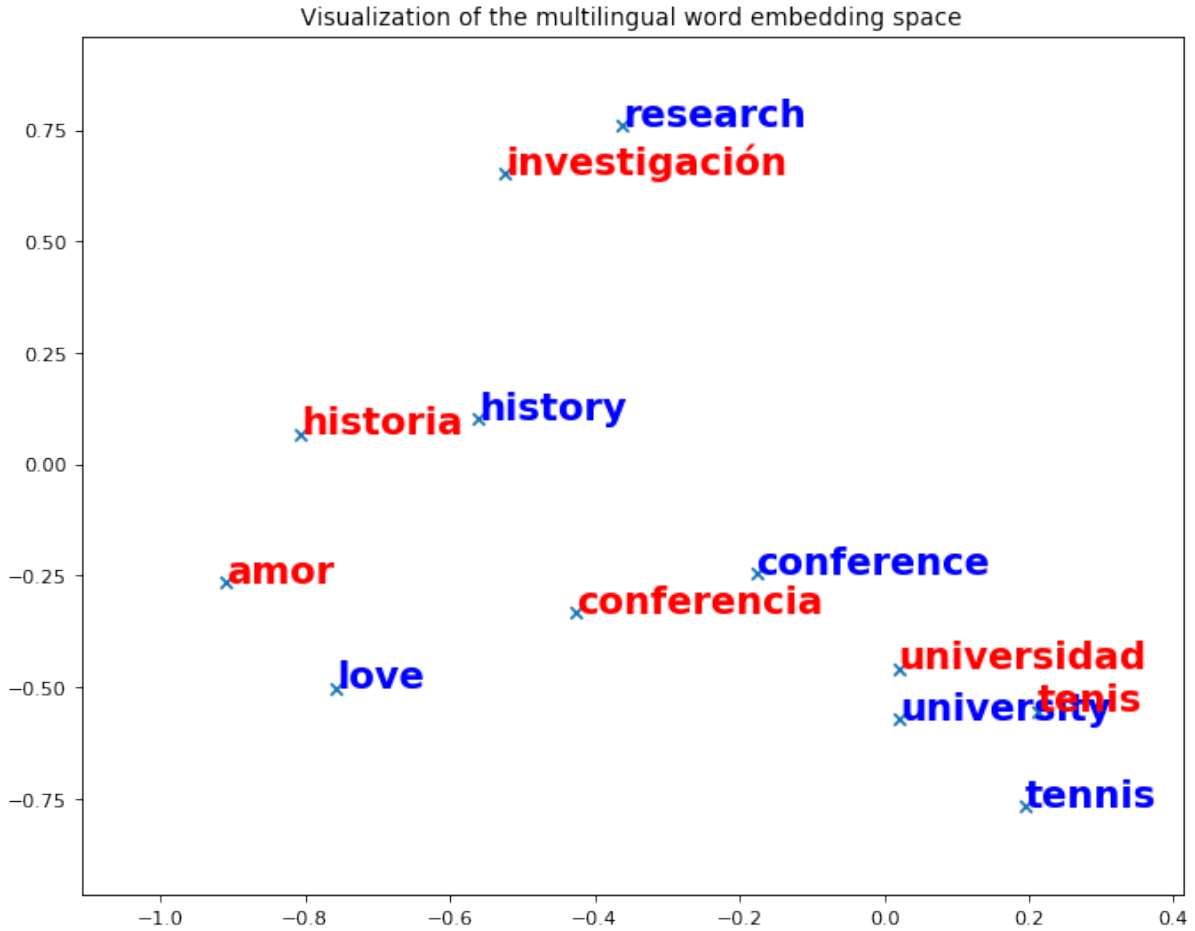


Figure 3.2: 2-dimensional projection of word embeddings in English and (aligned) Spanish.

KNN algorithm on texts represented as TF-IDF matrices of word unigrams. These neighbors are then encoded by the neural model, such that the concatenation of document encodings from both source and neighbor texts is used for predictions.

Furthermore, the labels and relative distances of the neighbor texts are also considered. The labels are two one-hot encoded vectors, representing the categories and sub-categories of the neighbor texts. These are then multiplied by the distance from source s to neighbor n texts, which is computed by the Minkowski distance, defined as:

$$d = \frac{\sum_i |s_i - n_i|^p}{1/p} \quad (3.8)$$

where the order p is defined as 1. Finally, these are averaged across all neighbors, resulting in one feature for categories, and another for sub-categories.

3.4 Summary

In this chapter, I proposed a deep learning architecture to perform cross-lingual classification of text. The base neural architecture is adapted from the Hierarchical Attention Network, modifying it by employing sparsemax attention layers and penalized hyperbolic tangent activation functions. Given the multilingual character of the dataset, words from texts are embedded using cross-lingual embeddings provided by MUSE. I also described the KNN augmentation, which is used to find neighboring texts. From these, we extract text to be embedded and encoded by the HAN, as well as one-hot encoded labels for categories and sub-categories to improve predictions.

Chapter 4

Experimental Evaluation

To assess the quality of the proposed architecture, training and testing was performed with examples from the Multilingual Folk Tale Database (MFTD). Section 4.1 introduces the dataset, while Section 4.2 explains the evaluation protocol. Then Section 4.3 discusses the obtained results, and Section 4.4 presents the visualizations of predictions. Finally, Section 4.5 summarizes the main ideas of this chapter.

4.1 The MFTD Dataset

The Multilingual Folk Tale Database (MFTD) is one of the largest online databases of folktales, containing thousands of texts in different languages. The documents used in our work were extracted from the MFTD website in order to build a dataset for training and testing, including labeled examples written in English, Portuguese, Spanish, French and Italian. This resulted in a total of 834 examples, as further detailed in Table 4.1.

All the texts from the resulting dataset are classified according to the ATU system. Due to the scarce amount of examples for each individual index, the first two levels from this hierarchy were used as labels, resulting in a single-label, cross-lingual, multi-class dataset with 7 categories and 42 subcategories.

4.2 Experimental Methodology

The experimental methodology used to assess the performance of the proposed classification model was based on cross-validation with a stratified k-fold partitioning algorithm using $k = 10$, and thus considering multiple splits of 90% and 10% for training and testing. This stratification was performed according to language-category labels, so that each split contains a balanced number of examples for all pairs of languages and categories. When comparing predictions and labels, the results were measured using micro and macro-averaged F_1 scores for ATU categories and sub-categories.

Table 4.1: Statistics for the MFTD dataset concerning labeled examples.

Property	EN	PT	ES	FR	IT	Total
Number of examples	341	67	186	108	132	834
Sentences per text (average)	63.9	47.9	54.1	61.9	55.3	58.8
Sentences per text (90th percentile)	133.0	94.4	100.0	125.9	101.7	122.0
Words per sentence (average)	29.6	29.4	27.4	26.0	24.1	27.8
Words per sentence (90th percentile)	57.0	59.0	52.0	50.0	43.0	53.0
Size of vocabulary	15,617	9,739	20,329	13,343	13,642	72,670
Distribution of examples						
C_0 : Animal tales	79	16	28	31	18	172
C_1 : Tales of magic	175	43	96	56	70	440
C_2 : Religious tales	8	1	14	5	8	36
C_3 : Realistic tales	14	3	12	3	7	39
C_4 : Tales of the stupid ogre	7	0	3	0	1	11
C_5 : Anecdotes and jokes	46	3	27	11	23	110
C_6 : Formula tales	7	1	6	2	5	26

In terms of model parameters, namely the dimensionality of model components, the embedding layer was composed of 300 units, while the GRU and attention layers consisted of 250 units. In terms of features, matrices of 100x50 word identifiers were used, in proximate conformity with the 90th percentiles of the MFTD dataset. Finally, training was performed in batches of 20 instances, for a maximum of 50 epochs and using the categorical cross-entropy loss function and Adam optimization algorithm [28]. To develop and train this model, the Keras¹ and scikit-learn² libraries were used.

To explain predictions, words from the source document were highlighted according to the attention weights of their corresponding annotations, while sentence attentions were displayed parallel to them.

4.3 Experimental results

Initial experiments were conducted with the English subset of the data. As baselines, I used a linear SVM with L2 regularization, inspired by the model from previous studies on the classification of folktales [5], and a KNN classifier with $k=2$, which was also used in the augmented neural model. Regarding the neural models, both the HAN leveraging only the text from the input document and the k -nearest neighbors augmented HAN (KNN-HAN) were used. As observed in Table 4.2, the neural models constantly outperformed the KNN baseline. Compared to the SVM, the base Hierarchical Attention Network achieves better performance regarding categories, but not for sub-categories. However, the augmentations in the model lead to a deterioration in performance.

With the complete multi-lingual dataset, we omit the baselines, as the differences in vocabulary for each language make them unsuited for cross-lingual classification. Once again, the augmented model did not show improvements for any metric when compared to the base HAN, as shown in Table 4.3. Details regarding the performance of each category are listed in Table 4.4, showing a tendency for

¹keras.io

²scikit-learn.org

Table 4.2: Evaluation results for the MFTD examples in English.

Model	Features	Categories		Subcategories	
		Micro F ₁	Macro F ₁	Micro F ₁	Macro F ₁
SVM	TF-IDF of word unigrams	0.710	0.282	0.365	0.233
SVM	TF-IDF of char [2,5]-grams	0.695	0.261	0.361	0.235
KNN	TF-IDF of word unigrams	0.581	0.188	0.193	0.099
KNN	TF-IDF of char [2,5]-grams	0.538	0.146	0.190	0.093
HAN	100x50 words, 250 units	0.712	0.358	0.245	0.149
KNN-HAN	100x50words, 250 units	0.648	0.313	0.179	0.090

Table 4.3: Evaluation results for the complete set of MFTD examples, in English, Portuguese, Spanish, French, and Italian.

Model	Features	Categories		Subcategories	
		Micro F ₁	Macro F ₁	Micro F ₁	Macro F ₁
HAN	100x50 word, 250 units	0.780	0.487	0.396	0.277
KNN-HAN	100x50 words, 250 units	0.768	0.471	0.290	0.170

worse performance in categories with fewer examples, namely C4, which had the least amount of texts. However, this tendency does not happen regarding performance across different languages (i.e., in some cases, languages with fewer examples end up achieving a higher performance), as shown in Table 4.5.

To study the impact of each feature in the cross-language classifier, ablation studies were performed where components or hyper-parameters were modified, as detailed in Table 4.6. These experiments indicate a favorable outcome from the sparsemax attention mechanism and choice of neural layer dimensions, as performance decreases when modifying them. However, the penalized hyperbolic tangent seems to have only improved the prediction of categories. Lastly, the KNN augmentation (which corresponds to the complete KNN-HAN architecture) resulted in deterioration of performance for all metrics, showing it was not successful at improving the base model.

4.4 Interpreting Results by Visualizing Attention Weights

Results can be explained, to some extent, by the relevance attributed to words and sentences during a prediction, i.e. the attention weights associated with their representations. To visualize predictions, I used examples from the MFTD dataset represented as 100x50 word identifiers, i.e. 100 sentence composed by 50 words each. When the model performs a prediction, it is possible to inspect the weights

⁴www.mftd.org/index.php?action=story&id=565

Table 4.4: F₁ scores for each category.

Model	C0	C1	C2	C3	C4	C5	C6
HAN	0.958	0.910	0.555	0.217	0.000	0.581	0.233
KNN-HAN	0.941	0.906	0.415	0.201	0.100	0.562	0.133

Table 4.5: F₁ scores concerning categories for each language.

Model	Metric	English	Portuguese	Spanish	French	Italian
HAN	Micro F ₁	0.766	0.764	0.759	0.850	0.805
KNN-HAN	Micro F ₁	0.756	0.793	0.743	0.817	0.791
HAN	Macro F ₁	0.425	0.572	0.509	0.737	0.625
KNN-HAN	Macro F ₁	0.431	0.660	0.510	0.617	0.588

Table 4.6: Ablation experiments for the HAN with multi-lingual examples from the MFTD dataset.

Model	Categories		Subcategories	
	Micro F ₁	Macro F ₁	Micro F ₁	Macro F ₁
HAN	0.780	0.487	0.396	0.277
Softmax Attention	0.748	0.422	0.363	0.236
Tanh Activation	0.764	0.449	0.410	0.282
KNN Augmentation	0.768	0.471	0.290	0.170
150 Units in Neural Layers	0.740	0.443	0.328	0.212

attributed to any layer, and as such obtain 100x50 attention weights for words, and 100 for sentences. Then, the words identifiers may be converted into the corresponding words, which are then used to generate an approximation of the source text.

By associating weights with words and sentences, we can highlight these accordingly. Using colors represented in the Hue Saturation Lightness (HSL) scale, the lightness value can be calculated as $L = 1 - C \cdot w_i$ to generate darker highlights for higher attention weights w_i , scaled by C to improve visual contrast. This simple calculation is possible due to the sparsemax function generating fewer positive weights than softmax, omitting the need of applying thresholds.

Given the word and color codes, then a web page is generated. To represent word attention, terms with non-zero positive word attention are highlighted with the appropriate color code. Furthermore, placing the cursor on a word displays its exact attention weights, to provide an increased precision. Sentence attention is displayed using a HTML table, where the first column corresponds to the *Sentence attention*, presenting a highlighted value, and the second column corresponds to the *Sentence* and its words. The values obtained from the output layer concerning categories are also displayed, to show the probability distribution over classifications and the predicted category, as shown in Figure 4.1.

From the obtained results for the overall corpus, there was a prevalence of higher attention weights for words related to the concepts of *animals* (e.g. goat, wolf) and *royalty* (e.g. king, princess). To study differences across categories, the sum of attention values for identical terms was considered for the English subset, and the words with highest values listed in Table 4.7. This listing demonstrates a clear connection between *Animal tales* and its most relevant words, which are very distinct from the other classifications. For other categories, such as *Tales of magic*, *Religious tales* or *Realistic tales*, there is a bigger overlap in relevant vocabulary, possibly due to the ubiquity of stories concerning royalty themes.

For other languages, the correspondent translations are often attributed similar weights, as exemplified in Figure 4.2.

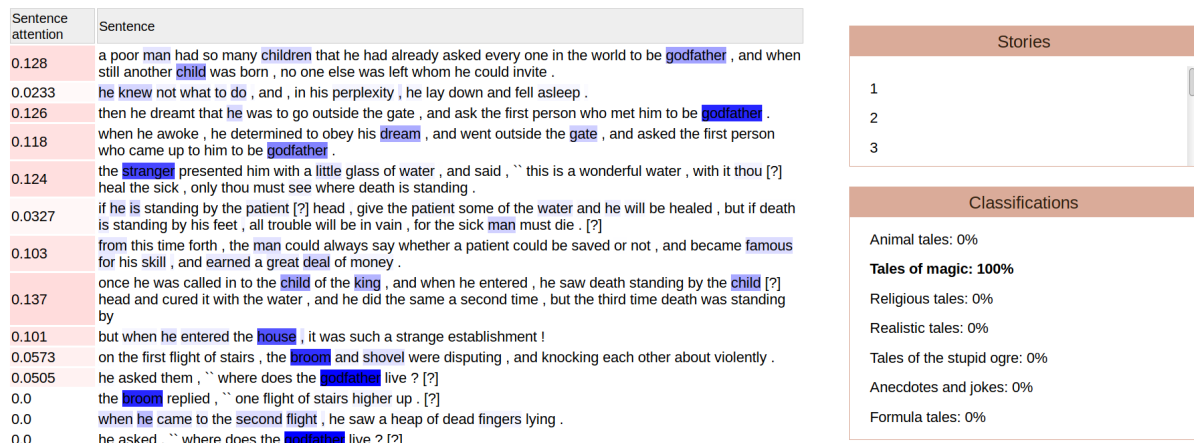


Figure 4.1: Web page generated for the tale *The Godfather*, originally written by Jacob and Wilhelm Grimm.⁴

Table 4.7: The 20 words with most total attention per category.

Category	Most relevant words
Animal tales	fox, wolf, cat, mouse, dog, man, ass, sparrow, lion, cock, turtle, hare, wood, bird, bear, master, goose, he, sheep, hen
Tales of magic	king, man, princess, daughter, lad, mother, father, girl, he, prince, she, old, woman, brothers, castle, son, child, bird, boy, brother
Religious tales	lord, king, father, tailor, shepherd, son, he, man, boy, peter, brother, merchant, woman, child, master, peasant, mother, saint, sword, flute
Realistic tales	king, princess, bride, prince, soldier, woman, maid, daughter, she, sweetheart, girl, father, bird, old, huntsman, son, man, beggar, lion, he
Stupid ogre	troll, tailor, lad, giant, peasant, devil, wood, old, maiden, woman, shepherd, bear, girl, boots, witch, man, child, roland, he, she
Jokes	man, king, lad, mother, cow, peasant, goody, hans, he, master, farmer, tailor, claus, boy, horse, old, goose, woman, frederick, she
Formula tales	cock, hen, mother, fox, pancake, sister, hare, little, pig, father, bride, tup, henny, horse, girl, dogs, woodcutter, shoes, birds, tree

An additional method of visualizing word attention can be produced with clouds of words. This methodology is popular in data visualization, and had been considered in previous work regarding using tf-idf and LRP values [20]. I explored this methodology for two purposes, using d3-cloud⁷. In one of them, a word cloud represents an individual story, where the attention weights of identical terms are summed and used to calculate the size of text, scaling linearly with the total attention. In the other approach, a word cloud is used to represent the stories belonging to a whole category, also by summing the attention weights for every identical terms. Both these approaches make it possible to visualize attention weights in a quicker observation, compared to text highlighting. Furthermore, they might be considered more visually appealing, as displayed in Figure 4.3, and also in Appendix A.

Concerning sentence attention, while it is harder to analyze their impact, higher attention values are often attributed to sentences containing words which are more relevant to intuition. Another apparent

⁶www.mftd.org/index.php?action=story&id=17

⁷github.com/jasondavies/d3-cloud

0.0 i [?] **wager** it is not in the almanac .

0.105 `` the **cat** [?] mouth soon again began to water for the delicious goods .

0.0936 `` all good **things** come in threes , [?] he said to the **mouse** .

0.098 `` i have been asked to be **godfather** again .

0.118 the **child** is totally black , only it has white paws .

0.104 [?] le **chat** ne tarda pas à se sentir de nouveau [?] à la bouche en pensai

0.0435 [?] jam **eux** sans trois , [?] [?] à la **souris** .

0.0 [?] on me **demande** de nouveau [?] le **parrain** .

0.0 [?] est tout noir , seules les **pattes** sont blanches , elles mis à part , il n ' :

0.165 un **enfant** comme ça ne nait [?] fois par siècle !

Figure 4.2: Highlighted excerpts for the English and French versions of the *Cat and Mouse Partnership* tale, originally written by Jacob and Wilhelm Grimm.⁶



Figure 4.3: Left: word cloud for the English version of the *Cat and Mouse Partnership* tale. Right: word cloud for English *Animal tales*.

difference is that attention weights appear to be less sparse for sentences than words, although both are calculated by similar sparsemax attention layers.

4.5 Summary

In this chapter, I presented some details regarding the experiments conducted. To train and test the model, I used texts extracted from the Multilingual Folk Tale Database, a single-label, cross-lingual dataset where two outputs were considered for training and testing (ATU categories and sub-categories). I also detailed the experiment methodology, which concerned a stratified k-fold approach to measure micro and macro-averaged F_1 scores, with stratification taking into account all combinations of language and categories, to provide balanced splits. I also detail some choices of hyper-parameters, namely the use of 100x50 word identifiers, and intermediate representations with 250 dimensions. Then I discussed the obtained results, where the neural architecture showed a superior performance compared to the baseline SVM method on the English subset of data. For the dataset regarding multiple languages, the

HAN and KNN augmented HAN were compared, where the first showed better results. By performing ablation studies, it was concluded that the sparsemax attention mechanism and increase in intermediate dimensions provided the best improvements to predictions, while other components had a mixed or negative impact. Finally, visualization of predictions were provided, using the attention weights of words and sentences to generate highlighted texts and clouds of words.

Chapter 5

Conclusions and Future Work

In this chapter, I present a closure to this dissertation, reflecting on the results achieved (Section 5.1), and presenting suggestions for future work (Section 5.2).

5.1 Summary of Contributions

In this M.Sc. thesis, I modified and extended the Hierarchical Attention Network (HAN) to classify folktales and generate visualizations to explain results. The modifications concerned the use of the penalized hyperbolic tangent as an activation function, and the use of an attention layer employing sparsemax weights. The extension regarded a KNN component, where similar texts are found using a KNN algorithm, and then their text encoded by the HAN and their labels connected to the output layers, in order to provide additional information for predictions.

Concerning an English subset of MFTD folktales, the baseline HAN model achieved the best performance for classifying categories, outperforming baselines inspired by previous research, but not for sub-categories. This might be explained due to the low number of examples present in most sub-categories, which is a common bottleneck for the training of neural networks. The model augmented with a KNN component did not show improvements for any metrics. Regarding the multi-lingual dataset, once again, the extended model did now show improvements of results, performing worse than the base HAN architecture.

By performing ablation studies with the multi-lingual dataset, the sparsemax attention mechanism showed the most impact, improving both classifications and visualizations. There were also positive results from increasing the dimensionality used in the intermediate representations of the model, while mixed results were obtained for the penalized hyperbolic tangent activation function, which improved predictions of categories but not of sub-categories. Concerning the KNN augmentation, it was not successful at improving the base architecture, showing a negative impact for all metrics.

Since performance tended to scale with the number of examples for a given category, this suggests that results would improve from more training data, namely on classes with fewer texts. On the other hand, even though the cross-lingual embeddings seem effective at aligning terms to their English trans-

lation, performance varied for each language, independently of the number or distribution of examples, possibly due to differences in the structure of each language.

The visualization of predictions by inspecting the sparsemax attention weights showed good results in a qualitative analysis. Few items were attributed high values, making it easier to find them in a text. The highlighting of words often provided the most relevant terms for a given category, and their translations were commonly highlighted as well in similar texts of other languages. Furthermore, word clouds were also explored, displaying relevant words of stories or whole categories in a visually appealing style.

5.2 Future Work

For future work, using pre-trained contextual representations [29] is an option that might improve results. There has also been recent research considering the hierarchical document structure [30, 31], which may be considered to improve this architecture. Additionally, other neural architectures could also show different results, such as the Transformer model [32], which is based solely on attention mechanisms.

Finally, there is also room for improving visualizations. One idea which was not explored was to rank the attention of words according to categories or classes, and present it in a word cloud. This could be useful to quickly understand how attention weights are distributed across categories and languages. Another idea that was not explored in the present paper concerned the KNN mechanism, which could be used to improve the interpretability of the model and help explaining predictions, e.g. listing neighbor texts. There are also other techniques which could be used to replace or extend the visualization of attention weights, such as finding important features with DeepLIFT [33].

Bibliography

- [1] J. Abello, P. Broadwell, and T. R. Tangherlini. Computational folkloristics. *Communications of the ACM*, 55(7), 2012.
- [2] T. Meder, F. Karsdorp, D. Nguyen, M. Theune, D. Trieschnigg, and I. Muiser. Automatic enrichment and classification of folktales in the Dutch folktale database. *Journal of American Folklore*, 129(511), 2016.
- [3] R. Järv. The singing wolf meets his kin abroad. Web-based databases of the Estonian folklore archives. *Studies in Oral Folk Literature*, 5, 2016.
- [4] A. Garcia-Fernandez, A.-L. Ligozat, and A. Vilnat. Construction and annotation of a French folkstale corpus. In *Proceedings of the International Conference on Language Resources and Evaluation*, 2014.
- [5] D. Nguyen, D. Trieschnigg, T. Meder, and M. Theune. Automatic classification of folk narrative genres. In *Proceedings of the Workshop on Language Technology for Historical Text*, 2012.
- [6] D. Nguyen, D. Trieschnigg, and M. Theune. Folktale classification using learning to rank. In *Proceedings of the European Conference on Information Retrieval*, 2013.
- [7] Y. Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57, 2016.
- [8] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.
- [9] F. Duarte, B. Martins, C. S. Pinto, and M. J. Silva. Deep neural models for ICD-10 coding of death certificates and autopsy reports in free-text. *Journal of Biomedical Informatics*, 80, 2018.
- [10] Z. Yang, D. Yan, C. Dyer, X. He, A. J. Smola, and E. H. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2016.
- [11] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the conference on empirical methods in natural language processing*, 2014.

- [12] M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2014.
- [13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [14] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- [15] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
- [16] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [17] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.
- [18] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine learning*, 47(2-3).
- [19] R. Johnson and T. Zhang. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2017.
- [20] F. Horn, L. Arras, G. Montavon, K.-R. Müllerf, and W. Samek. Exploring text datasets by visualizing relevant words. *arXiv preprint arXiv:1707.05261*, 2017.
- [21] L. Arras, F. Horn, G. Montavon, K.-R. Müller, and W. Samek. Explaining predictions of non-linear classifiers in NLP. *arXiv preprint arXiv:1606.07298*, 2016.
- [22] L. Arras, F. Horn, G. Montavon, K.-R. Müller, and W. Samek. "What is relevant in a text document?": An interpretable machine learning approach. *PLoS ONE*, 12(8), 2016.
- [23] A. Martins and R. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the International Conference on Machine Learning*, 2016.
- [24] S. Eger, P. Youssef, and I. Gurevych. Is it time to swish? comparing deep learning activation functions across NLP tasks. *arXiv preprint arXiv:1901.02671*, 2019.
- [25] Z. Wang, W. Hamza, and L. Song. k -nearest neighbor augmented neural networks for text classification. *arXiv preprint arXiv:1708.07863*, 2017.
- [26] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

- [27] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.
- [28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [30] Y. Mao, J. Tian, J. Han, and X. Ren. End-to-end hierarchical text classification with label assignment policy. In *International Conference on Learning Representations*, 2019.
- [31] S. Baker and A. Korhonen. Initializing neural networks for hierarchical multi-label text classification. In *Proceedings of Workshop on Biomedical Natural Language Processing*.
- [32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the Annual conference on Neural Information Processing Systems*.
- [33] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *Proceedings of the International Conference on Machine Learning*.

Appendix A

Word clouds for categories

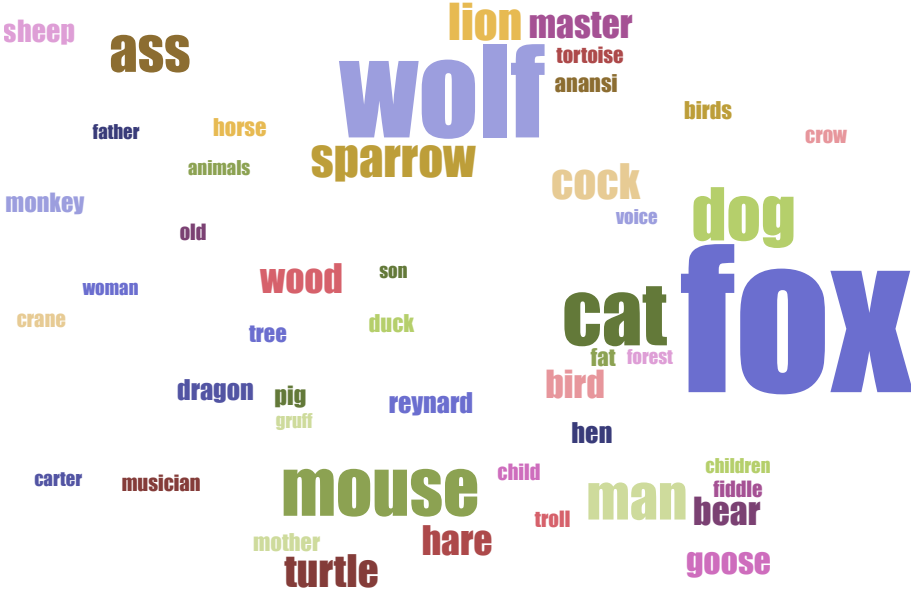


Figure A.1: Word clouds for the category Animal tales.

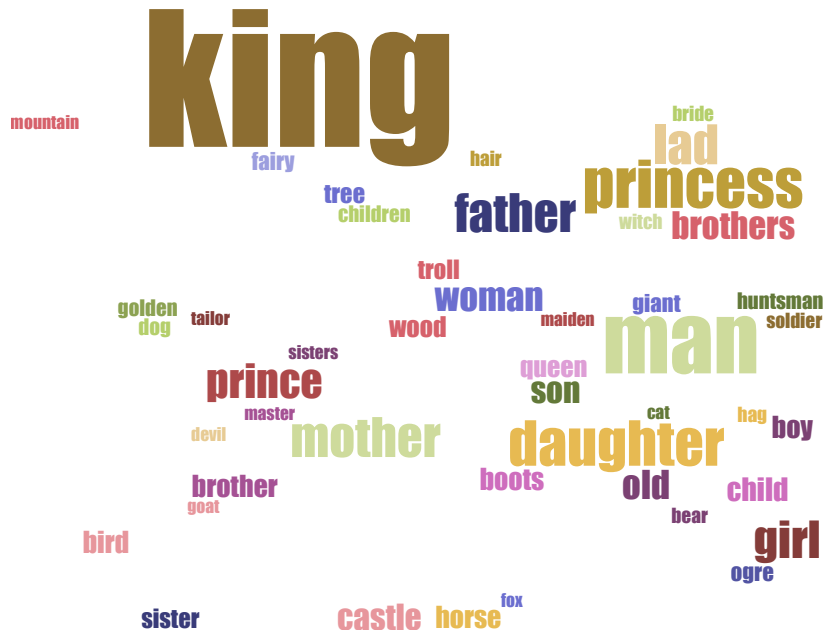


Figure A.2: Word clouds for the category Tales of magic.



Figure A.3: Word clouds for the category Religious tales.

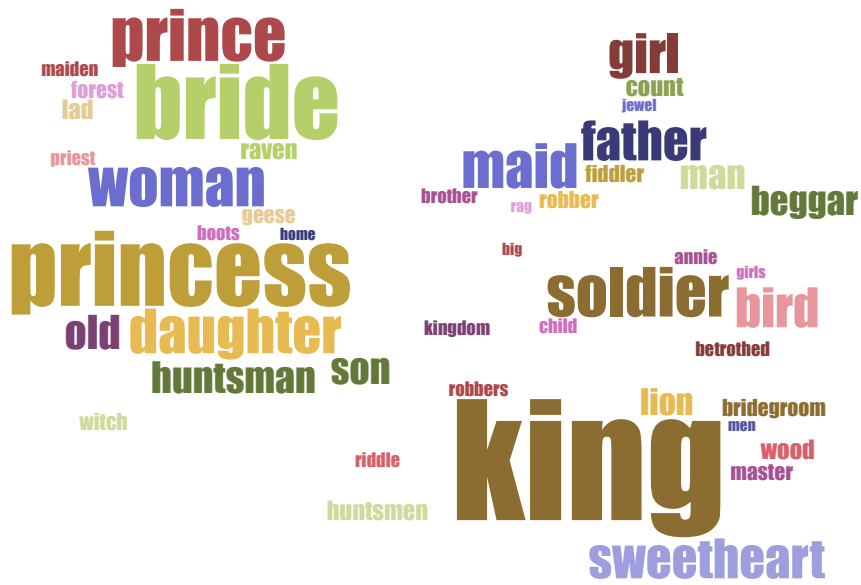


Figure A.4: Word clouds for the category Realistic tales.



Figure A.5: Word clouds for the category Tales of the stupid ogre.

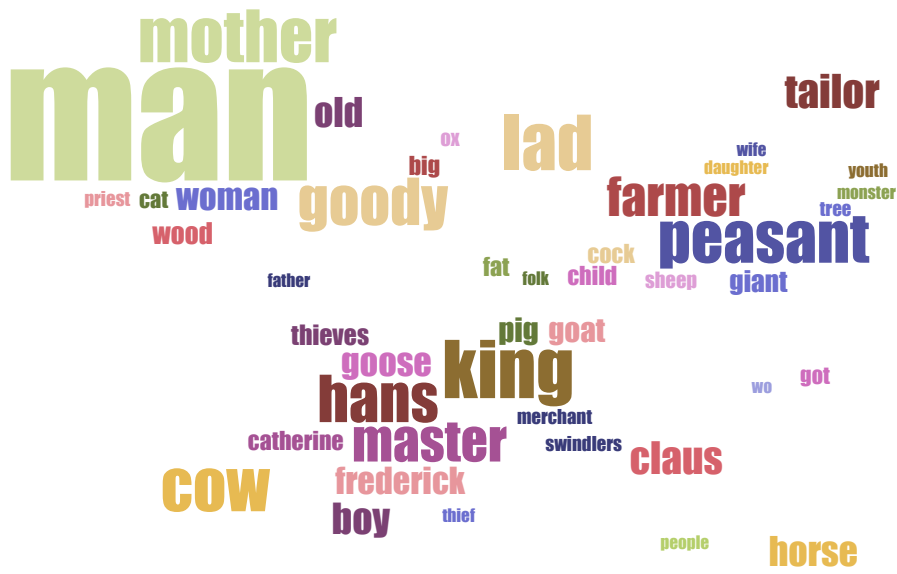


Figure A.6: Word clouds for the category Anecdotes and jokes.



Figure A.7: Word clouds for the category Formula tales.