university of groningen

faculty of science and engineering

UNIVERSITY OF GRONINGEN, THE NETHERLANDS

A MASTER THESIS ON:

Uncertainty in Semantic Language Modeling with PIXELS

BY:

Ștefania Radu

Internal Supervisors:
Dr. Matias Valdenegro-Toro,
Dr. Marco Zullich

Abstract

Traditional Language Models like BERT are trained using raw text, split into separate chunks using a tokenizer. These models suffer from 3 main challenges: a lack of context understanding, a bottleneck in the vocabulary, and unreliable predictions caused by high epistemic uncertainty. This study investigates a new approach – Visual Language Models (VLMs) – by rendering text as an image and replacing the masked language modeling task with patch reconstruction at the pixel level. The novelty of this work consists of analysing uncertainty and confidence in VLMs models across 18 languages and 7 scripts, all part of 3 semantically challenging tasks: Named Entity Recognition (NER), Sequence Classification (SC), and Question-Answering (QA). This is achieved through several Uncertainty Quantification methods such as Monte Carlo Dropout, Transformer Attention, and Ensemble Learning. The results suggest that VLMs underestimate uncertainty when reconstructing patches, especially when a large proportion of the image is masked. The uncertainty is also influenced by the script, with Latin languages displaying lower uncertainty, compared to the Geez or Chinese Characters scripts. The findings on ensemble learning show better performance when applying hyperparameter tuning during the NER and QA tasks across 16 languages, as well as improved overall calibration.

Contents

1 Introduction

1.1 Prologue

"The limits of my language mean the limits of my world."

—Ludwig Wittgenstein

One of the many implications of this statement written by Ludwig Wittgenstein in the *Tractatus Logico-Philosophicus* (Wittgenstein, 1922) is that language defines the framework through which, we humans, perceive and interpret the world. In other words, language shapes the way we think, and by that, it changes who we are. To understand and produce language, humans have evolved specialized brain regions, as well as sophisticated language-learning mechanisms. The most important areas are Broca's area, involved in speech production and grammar processing, and Wernicke's area, which is essential for understanding spoken language and semantic processing. The existence of a neurological language acquisition device was proposed by Chomsky (1965), who believes that children are born with the ability to extract underlying grammar rules from language. Interaction with the environment plays a crucial role in the learning process, as well. Children experiment actively with the language and use techniques like imitation or positive reinforcement (Skinner, 1957), which enables them to gradually understand the rules of syntax and grammar through trial and error. While imperfect and prone to errors, this approach is effective nonetheless, at least as far as humans are concerned. When it comes to machines, teaching them how to communicate in the same way we do has been a difficult challenge ever since the Turing test was introduced in 1950.

Advanced natural language modeling (NLP) methods were developed only decades later. Statistical language models (Jelinek, 1998; Rosenfeld, 2000) originated in the 1990s and rely on next-word prediction given a context. Neural language models apply neural networks to encode the probability of word sequences (Bengio et al., 2000). In this context, static word representations were then used to create models such as word2vec (Mikolov et al., 2013), which proved to be very effective at learning text features that generalize across a wide range of NLP tasks. In an attempt to capture context-aware information, pretrained language models became very popular in 2018. Based on the transformer architecture (Vaswani et al., 2017) and the self-attention mechanism, models like BERT (Devlin et al., 2018) can learn general-purpose semantic features during pretraining and be finetuned on specific downstream tasks later. The transferable nature of these models makes them very versatile. To improve their applicability in real-world tasks where the quantity of text data is large, scaling the model size and the dataset size is what turned pretrained models into Large Language Models (LLMs), that are often considered general-purpose task solvers (Brown et al., 2020).

After the release of ChatGPT in 2022, the number of papers published every day on the topic of LLMs has increased more than 20-fold (Zhao et al., 2023). The number of parameters in these models jumped from 340 millions in implementations such as BERT (Devlin et al., 2018) to billions of parameters in models like GPT-3 (Brown et al., 2020) or LLaMA (Touvron et al., 2023). Despite their obvious popularity, LLMs suffer from

many limitations, some of which will be discussed in more detail in the next chapter. It is well known that these models require large quantities of training data and huge computational resources (Abadi et al., 2016), often available only to the industry. This means that many training details are not revealed to the public. Another critical issue that came to light shortly after ChatGPT was made available is that of hallucinations (OpenAI, 2023). This means that during the text generation process, LLMs can produce incorrect, nonsensical and even harmful answers. However, hallucinations are inherent to traditional LLMs, considering that they are optimized to generate natural-sounding language, which does not necessarily entail factual certainty (methods like reinforcement learning from human feedback (Christiano et al., 2017) have been shown to reduce this problem). LLMs often do not express uncertainty and display a high level of confidence in their responses, regardless of their accuracy – *the uncertainty problem*. This can mislead users into trusting incorrect or unverified information. For this reason, it is difficult to engineer models that can utilize the relevant surrounding tokens to perform well on semantic tasks, such as word-level or sentence-level understanding – *the semantics problem*. A related limitation refers to the finite number of categorical inputs that LLMs can support – *the vocabulary problem*.

The next section will first give an overview of language modeling and traditional LLMs, including how they work and common applications. It will then discuss the previously mentioned challenges: the vocabulary problem, the context problem and the confidence problem. Finally, a new type of approach – visual language models – will be introduced, which is here to solve the limitations of traditional LLMs.

1.2 About This Thesis

This work diverges from traditional language models, as presented in Section 2.1, and takes a different path towards visual language models, by incorporating image-based methods, such as MAE and ViT (Section 2.2). The PIXEL model introduced by Rust et al. (2022) serves as a starting point for this thesis. The main aim is to study uncertainty in visual language models focusing on semantic tasks. This will be achieved by addressing the three challenges observed by the author: The Semantic Problem, The Vocabulary Problem, and The Uncertainty Problem, as illustrated in Figure 1.2.1. Given the challenging nature of semantic processing and the fewer studies dedicated to it, this research will center on finetuning models to solve tasks like named entity recognition, sequence classification, and question answering. Solving the vocabulary bottleneck of traditional language models which rely on a close vocabulary can be achieved by using pixel-based models which do not require a fixed vocabulary. Finally, to tackle the uncertainty problem, this work will make use of existing techniques for quantifying uncertainty, and apply them to visual language models, which also represent the biggest novelty of this study. This includes uncertainty quantification at the pixel level using Monte Carlo methods, ensemble learning applied to models finetuned on three semantic tasks across 19 languages, but also an analysis of the attention mechanism in visual language models, and an attempt to pretrain a multilingual model with improved contextual understanding. For a more detailed overview of these issues in the literature, refer to Section 2.

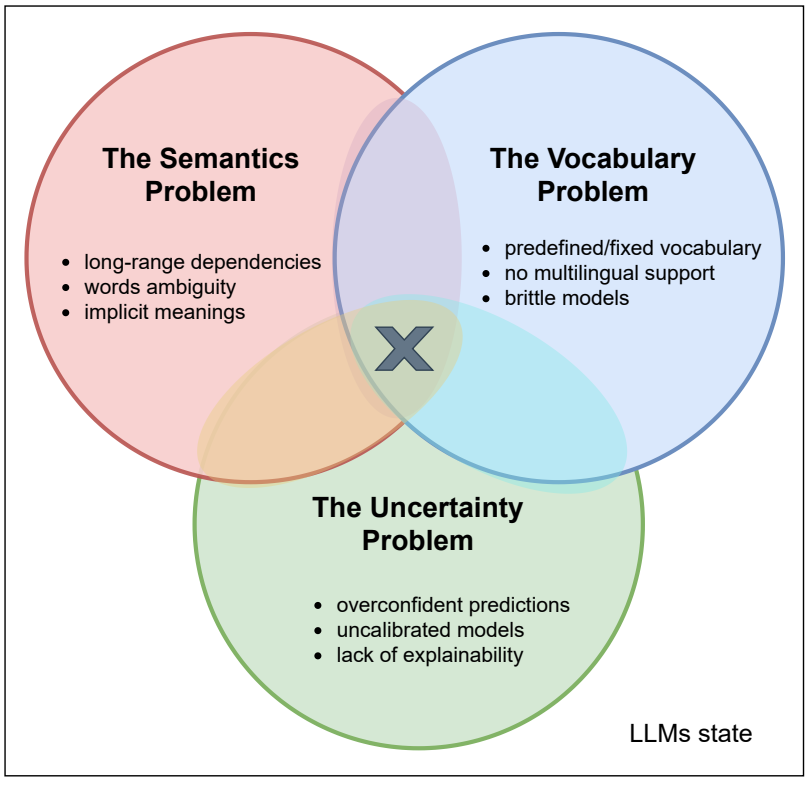
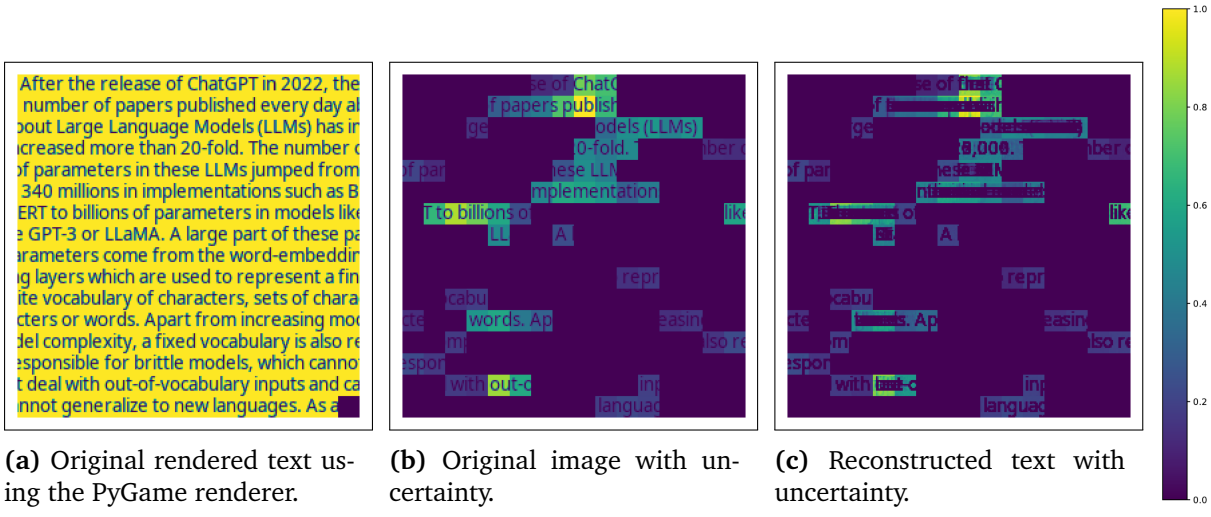


Figure 1.2.1: The current work aims to study three of the (many) existing problems in the current LLMs landscape: The Semantics Problem (Section 2.4), The Vocabulary Problem (Section 2.5), and The Uncertainty Problem (Section 2.6). Here, **X** marks the spot representing the research area that this thesis falls under.

The research questions to be addressed are:

1. Do visual language models represent a viable solution for the semantic processing of text?
2. How can uncertainty quantification and calibration methods be integrated into visual language models?
3. How do visual language models encode uncertainty at the pixel level?
4. How is the attention mechanism represented in visual language models?
5. What is the effect of ensemble learning in finetuning visual language models?

The remainder of this thesis is structured as follows: Section 2 will present the Literature Review and motivation for this work in the form of three main challenges. Next, Section 3 describes the Methods, including the data used, the model architecture, and the training process. Section 4 outlines the Experimental Setup, followed by a presentation of the Results in Section 5. Finally, Section 6 will discuss the results and give a conclusion.



(a) Original rendered text using the PyGame renderer.

(b) Original image with uncertainty.

(c) Reconstructed text with uncertainty.

Figure 1.2.2: Example of uncertainty quantification at the patch-level for an image containing text from the proposal of this thesis. The uncertainty is measured as the standard deviation for each patch after Monte Carlo Dropout. Brighter colors indicate more uncertainty.

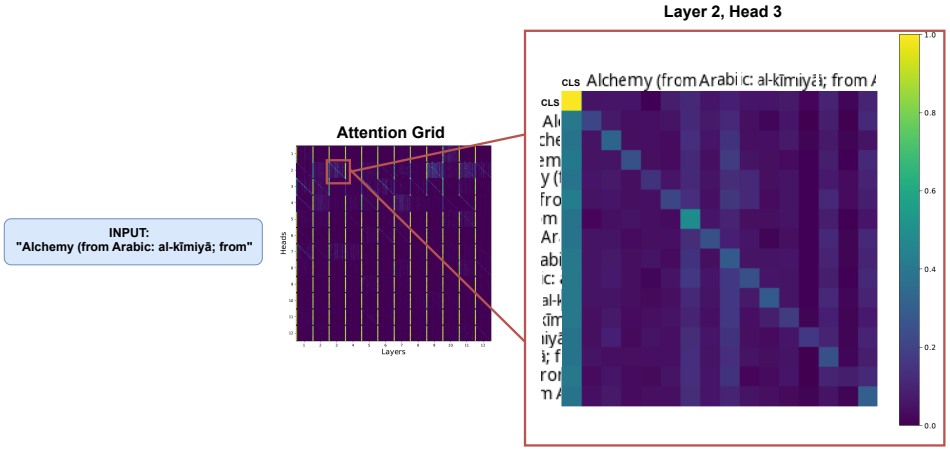


Figure 1.2.3: Model-level (attention grid) and neuron-level (layer 2, head 3) views of attention in the PIXEL model for a short input text from the English Wikipedia. The attention grid contains 12 attention layers with 12 attention heads each. Each patch in the attention cell attends to the other patches in the sequence according to the dot product between the query (of the *attender* patch) and the key (of the *attended* patch).

2 Literature Review

2.1 Traditional Large Language Models

Tokenization The majority of Large Language Models (LLMs) use raw text data as input, which requires preprocessing. These steps usually involve applying tokenizers and embedding layers. During tokenization, the text is divided into smaller chunks called *tokens* (Jurafsky & Martin, 2023), that range from individual characters to words or phrases. More advanced tokenizers include Byte-Pair Encoding (BPE) (Sennrich et al., 2015), WordPiece (Devlin et al., 2018), and SentencePiece (Kudo & Richardson, 2018).

BPE is a compression algorithm proposed by Sennrich et al. (2015), which learns a tokenization by iteratively replacing pairs of adjacent symbols with a new symbol representing that pair. It does this after the desired vocabulary size has been reached. WordPiece behaves similarly, with the main difference being that it does not choose the most frequent symbol pair, but the one that maximizes the likelihood of the training data once added to the vocabulary. Unlike BPE and WordPiece, the Unigram tokenizer initializes its base vocabulary to a large number of symbols and removes symbols progressively to obtain a smaller vocabulary. This is achieved using a log-likelihood loss (Equation 2.1.1), defined over the training data, given the current vocabulary and a unigram language model, where $x_1 \dots x_N$ represent words and the set $S(x_i)$ contains all possible tokenizations for word x_i. Unigram is usually used in combination with SentencePiece and it does not rely on the assumption that words are separated through spaces, which is the case for Chinese or Japanese. It treats the input as a raw input stream, which makes it applicable in a multilingual context. The output of the tokenizer serves as input for the first layer of the transformer model – the embedding layer.

$$L = -\sum_{i=1}^{N} \log\left(\sum_{x \in S(x_i)} p(x)\right) \tag{2.1.1}$$

Embeddings The embedding layer is then used to encode tokens (word-embeddings) or sequences of n-tokens (n-grams embeddings), by projecting them into a dense vector representation from a continuous space (Bengio et al., 2000). Since this can also be framed as a problem of maximizing mutual information (Kong, 2019), semantically similar tokens will also have similar representations in the dense space. This allows the model to process text in a way that reflects the underlying meanings of words and phrases and generalize across contexts. Machine learning algorithms like TF-IDF (Spärck Jones, 2004) are widely used in information retrival tasks and use statistical measures such as term-frequency and inverse term frequency to find relevant words in text, given a corpus. Predictive models like word2vec (Mikolov et al., 2013) learn embedding representations through predicting word w by maximizing the probability $p(w|C)$, given the context C. Several types of embeddings can also be combined to improve performance. For instance, BERT (Devlin et al., 2018) summed token embeddings with segment and positional embeddings, which provide information about the position of each token within the sequence.

The Attention Mechanism The text processing abilities of Large Language Models (LLMs) are based on the transformer architecture, introduced by Vaswani et al. (2017).

While previous architectures process text sequentially (Gehring et al., 2017), transformers use the attention mechanism, consisting of three main components: *queries* (Q), *keys* (K) and *values* (V). During the first step, the dot product is used to compute a score for each query, given a key. Then, the scores are passed through a `softmax` function to generate the weights. Finally, the attention is computed by a weighted sum of the value vectors, where each value vector is paired with a corresponding key. The complete computation is shown in Equation 2.1.2, where d_k is the dimension of K. In the transformer model, multiple attention functions are performed in parallel, known as multi-head attention. This mechanism enables the model to attend to information from different subspaces and positions.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1.2}$$

Training Training a language model involves two phases: pretraining and finetuning. During pretraining, the model has access to a diverse corpus of text data. pretraining typically utilizes self-supervized learning, that does not require labels. Models like GPT use a next-word prediction task where the next word in a sequence has to be predicted, while BERT combined two pretraining tasks: Masked Language Modeling (MLM) – where random words are masked from the input text and the model learns to predict them, and Next Sentence Prediction (NSP). During finetuning, the model is trained on a smaller, more specific dataset. In this phase, the weights are adjusted to specialize on a particular task.

Evaluation There are a variety of tasks used to test the capabilities of language models, including syntactic and semantic tasks. Examples of syntactic tasks are part-of-speech (POS) tagging and dependency parsing using data from Universal Dependencies v2.10 treebanks (Nivre et al., 2020). Semantic processing requires the model to understand word-level information through tasks like Named Entity Recognition (NER) or sentence-level information through tasks like question answering, assessed by the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016). The General Language Understanding Evaluation (GLUE) (A. Wang et al., 2018) dataset is another popular benchmark, including sentiment analysis, textual entailment, and document similarity. Despite recent advancements, the semantics problem remains a recurrent challenge.

2.2 Learning Through Masking

Masked Language Modeling The concept of masking and reconstructing parts of the data has shown great potential in both NLP and computer vision (CV) related tasks. *Masked language modeling* (MLM) used in models such as BERT (Devlin et al., 2018) involves holding out a part of the input and training the model to predict it while having access to the left and right context. These models are called bidirectional autoencoders and can perform more advanced linguistic tasks, such as labeling or parsing, as they allow the self-attention mechanism to range over the entire input (Jurafsky & Martin, 2023). MLM is a common pertaining task because it enables the model to form useful representations about the meaning of words, by mapping input embeddings $(x_1 \dots x_n)$

to contextualized embeddings of the same length $(y_1 \dots y_n)$.

During training, a random sample of tokens from each training example is selected to be part of the learning task. Each token can then be part of one of three categories based on a set of probabilities. It can be replaced with the `[MASK]` token, replaced with another token from the vocabulary, or left unchanged. In the case of BERT, out of 15% of tokens selected for learning, 80% are replaced with `[MASK]`, 10% are replaced with randomly selected tokens, and the remaining 10% are left unchanged. While the proponents of BERT suggest that 15% represents a sweet spot between good representations and learning efficiency, other studies argue that the exact number is not universal, but rather depends on different factors, like model size or training strategy (Wettig et al., 2022). The findings suggest that masking 40% of the input (as opposed to 15%) leads to models that perform better when finetuned on GLUE and SQuAD (Izsak et al., 2021). Moreover, even with a masking rate of 80%, the learned representations still preserve up to 95% of the performance in downstream tasks (Wettig et al., 2022). This is to show that there is not yet a clear-cut answer about the ideal masking rate to be used in MLM.

Vision Transformer Inspired by the language transformer (Vaswani et al., 2017), Dosovitskiy et al. (2020) proposed the vision transformer model as an image-based pipeline. Instead of text tokenization, the input to the ViT is an image, which needs to be split into fixed-sized patches of pixels. These patches are linearly embedded and then encoded by a traditional transformer encoder. One of the advantages of transforming the image into a sequence of patches is that no additional image-specific inductive biases are introduced into the architecture. Variants of ViT are successfully used in CV tasks, such as object recognition (Touvron et al., 2021) or segmentation (Liu et al., 2021), and they serve as backbones for various other models, like Masked Autoencders (He et al., 2022).

Masked Autoencoder In the context of computer vision, masked image encoding works similarly to MLM, by masking regions of an image and then learning to reconstruct the whole image. These models have an encoder-decoder architecture and are referred to as masked autoencoders (MAE) (He et al., 2022). By following the autoencoder mechanism, the MAE reconstructs original images given some partial observations. During masking, the image is divided into non-overlapping patches and masked using random sampling. The masking rate used by the MAE is 75%, which is considered high, but it is motivated by the need to learn useful representations of larger elements, such as objects or scenes. The ViT encoder is applied on the visible patches only and creates embeddings using a linear projection. Additional positional embeddings are added and the output is fed to a series of transformer blocks. The input to the decoder consists of both the visible and masked patches. The model will reconstruct the patches by predicting a value for each pixel in the masked patch.

2.3 Text As Visual Representation

It is clear that language models feed on language in order to deliver their impressive results, but the way in which language should be *represented* remains a matter of debate. As discussed in Section 2.1, traditional LLMs rely solely on text and use tokenizers to split sequences of text into smaller units, referred to as tokens. A different approach is to

Phenomena	Word	BPE (5k)	
Vowelization	كتاب	كتاب	(1)
	الكِتابُ	الك · ِ · ت · اب · ُ	(5)
Misspelling	language	language	(1)
	langauge	la · ng · au · ge	(4)
Visually Similar Characters	really	really	(1)
	rea11y	re · a · 1 · 1 · y	(5)
Shared Character Components	확인한다	확인 · 한 · 다	(3)
	확인했다	확인 · 했다	(2)

Figure 2.3.1: Examples of linguistics phenomena that can alter the representation of text in subword models. Image from Salesky et al. (2021).

focus on the visual representation of words, as opposed to the text-based representation. This method is tokenization-free and it does not require a vocabulary, making it very robust against noise. Unlike byte-based models, which depend on observed sequences and can be susceptible to variations that visually appear very similar (Figure 2.3.1, Salesky et al. (2021)), visual language models encode similarities between characters, which is more consistent with the idea that humans perceive text visually, and not from unicode representations (Salesky et al., 2021).

The first study to use visual features of text in order to create embeddings was applied to Chinese and used linearizing bitmaps of characters or words (Aldón Mínguez et al., 2016). By using shared character components from Chinese or Korean, it becomes easier to generalize to new and less frequent characters. Different studies (Dai & Cai, 2017; Sun et al., 2018; Salesky et al., 2021) used rendering techniques to obtain images of text. In this context, text rendering involves converting character codes into glyph indices, which are then used to generate the corresponding glyph images, while applying various styles, fonts, sizes, and colors. A glyph often contains one character only, but it can also represent accents or multiple characters in languages where ligatures are common, like Arabic. Dai & Cai (2017) used text rendering in Chinese, Japanese, and Korean, and extracted visual features from a Convolutional Neural Network (CNN) to perform text classification. Similarly, Sun et al. (2018) applied convolutions to squared rendered images to perform sentiment analysis in Chinese and English.

In the context of machine translation, Salesky et al. (2021) suggested a very robust approach based on a variation of the ViT. The training data is rendered into gray-scale images using the Pygame backend and a slicing window is applied to create patches, which act as tokens. Then, a 2D convolutional block followed by linear projection is used to create embeddings, which serve as input for the transformer encoder. The translation happens directly from pixel representations, without any word preprocessing. After training on seven language pairs, the approach matches the performance of traditional language models, with additional advantages. It is more robust to character permutations or substitutions, and it does not rely on text preprocessing steps, such as

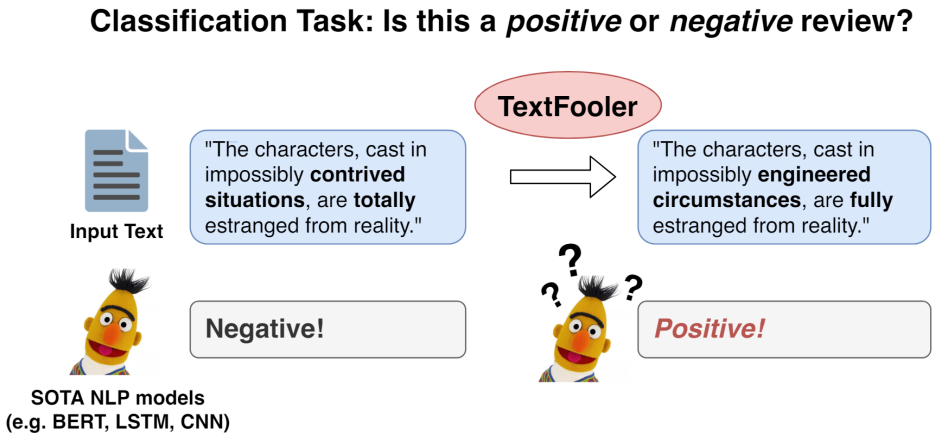


Figure 2.4.1: Example of instance when BERT is fooled during a sentiment classification task. Replacing some words with synonyms can completely change the outcome of the prediction. Image from Jin et al. (2020).

tokenization or segmentation.

Inspired by Salesky et al. (2021), Rust et al. (2022) proposed a more extensive model called *Pixel based Encoder of Language* or PIXEL, which aims to transform language modeling into a visual recognition task. PIXEL does not rely on a predefined vocabulary and it is trained to reconstruct missing patches of text, by following a ViT-MAE architecture. Similarly to BERT, the model is pretrained on rendered versions of the Wikipedia and BookCorpus datasets and it is evaluated on 32 topologically diverse languages, across 14 scripts. The results indicate that PIXEL outperforms BERT in syntactic tasks, such as part-of-speech tagging and dependency parsing while using fewer parameters. The model also demonstrates robustness to orthographic attacks and linguistic code-switching. However, during semantic tasks like named entity recognition, sequence classification, and question answering, PIXEL is struggling to retain semantic knowledge and transfer it across scripts. Reasons for this might include a lack of multilingual pretraining, as well as a limited ability to capture contextual information due to the use of unigram patch embeddings. While raw performance is desirable, it is crucial to have models that are reliable and explainable. Fortunately, uncertainty and confidence quantification methods are here to help achieve these goals.

2.4 The Semantics Problem

To evaluate LLMs, research usually relies on a combination of syntactic and semantic tasks. However, the focus is generally on syntactic tasks, and in models such as BERT, the studies on how knowledge is processed in semantic tasks are more challenging and therefore scarce (Rogers et al., 2021). This happens for several reasons, including the ambiguity of words, long-range dependencies between tokens, world knowledge, and implicit meanings. Addressing the semantics problem requires a deeper understanding of the linguistic context, as opposed to syntactic tasks.

Evidence suggests that language models encode semantic information about semantic roles, entity types and relations (Tenney, Xia, et al., 2019), but it is still unclear how

the knowledge is encoded into the model weights. In NER tasks, where the goal is to assign labels to words, such as person names, organizations, locations, etc., LLMs fail to account for entity replacements, despite showing high F1 scores (Balasubramanian et al., 2020). In classification tasks, slightly changing the words can alter completely the prediction result (Figure 2.4.1, Jin et al. (2020)). This suggests that the features learned by the model are not general enough to support a real understanding of named entities from sentences (Tenney, Das, & Pavlick, 2019). In the case of word knowledge, models have trouble reasoning based on their acquired knowledge. For example, BERT can learn various object attributes, but it cannot make inferences about the relationships between these attributes (Forbes et al., 2019). In addition, Adelani et al. (2021) indicates that the performance in downstream tasks decreases with the number of inference steps.

What makes semantic tasks significantly more difficult is the need to maintain context over longer text spans. This is because relevant information about a word cannot always be found in its proximity. For instance, in sequence classification tasks where the goal is to extract a sentiment or thematic content from an entire document, the relevant cues may be distributed across the text. The most common solution to tackle the long-term dependency problem is by employing self-attention, allowing each token in the input sequence to dynamically attend to all other positions. Nevertheless, this is not always sufficient for large contexts.

Different approaches emerged in the literature to account for larger contexts. When developing the T5 model, Raffel et al. (2020) proposed a new pretraining task that integrates more information and involves predicting a missing segment of text given the surrounding context. Additionally, the authors used a span-based masking strategy – rather than random masking – where contiguous random spans of tokens are replaced with a single mask token, thus increasing the length of the segment to be predicted. To boost memorization capabilities and context understanding, models like CANINE-C (Clark et al., 2022) included n-gram embeddings, which improved performance in NER tasks. Given the challenging nature of semantic tasks, this thesis will focus on investigating these aspects.

2.5 The Vocabulary Problem

A large number of language models use a closed-vocabulary approach when representing language. This means that the number of tokens in the vocabulary $|V|$ is finite, leading to a bottleneck when one tries to expand it with new words. Vocabularies are typically constructed using tokenization methods, such as BPE (Sennrich et al., 2015), WordPiece (Devlin et al., 2018) or SentencePiece (Kudo & Richardson, 2018). Their goal is to preprocess the text by cleaning and transforming it into a list of tokens. While traditional vocabularies can be a viable solution for monolingual contexts, tokenization becomes a bottleneck in multilingual settings, when trying to scale up the number of languages (Conneau et al., 2019).

One of the main issues with a finite vocabulary is the occurrence of out-of-vocabulary (OOV) words – words that were seen very rarely during training or not at all (Mielke et al., 2021). When a word is not present in the predefined vocabulary, it must be decomposed into smaller subword units, resulting in a loss of information. Current solutions to the vocabulary problem include subword tokenization and extending the closed vo-

cabulary to an open vocabulary.

Subword tokenization involves splitting the words-like tokens into smaller units, including all characters from the training set (Mielke et al., 2021). While using subwords can improve the processing of morphologically rich languages (Mikolov et al., 2012), it generally comes at the cost of efficiency (Sennrich et al., 2015). This tokenization process can lead to longer sequences of tokens, increasing the computational load of the model. Moreover, it makes the models more susceptible to noise, such as misspellings (Eger et al., 2019) or character swaps (Belinkov & Bisk, 2017), since small variations in text can result in different token sequences. Moreover, subword tokenization also introduces an additional layer of complexity, by requiring the reconstruction of the original meaning from smaller units (Bostrom & Durrett, 2020).

A different way of tackling the closed vocabulary is extending it so that models can generate new words. Mielke & Eisner (2019) proposed an approach based on a recurrent neural network language model, which regularizes word embeddings to be predictive of their spelling and predicts unknown words dynamically. Kawakami et al. (2017) improved this model, by adding a cache feature to remember predicted words. It is also possible to expand the vocabulary after training through post-hoc methods like transliteration (Moosa et al., 2022) or subword mapping (Vernikos & Popescu-Belis, 2021), but these do not represent permanent solutions.

The vocabulary problem also poses significant challenges for multilingual language models. Supporting multiple languages requires a larger vocabulary to cover diverse linguistic features and scripts, which is often impractical within the constraints of a fixed vocabulary size. Wu & Dredze (2019) noted that multilingual models struggle with resource allocation across languages, leading to suboptimal performance in less represented languages, during tasks like named entity recognition, part-of-speech tagging, and dependency parsing. Furthermore, imbalanced vocabulary representation can exacerbate biases, resulting in unfair treatment of certain languages (Wan, 2021). The trade-off in vocabulary allocation means that models either inadequately represent some languages or become too large in size and computational requirements. The current work addresses the vocabulary bottleneck problem, with the aim of advancing the inclusivity of language models, which are often dominated by commonly used languages.

2.6 The Uncertainty Problem

Handling uncertainty is an open problem in NLP, especially when it comes to very complex language models with billions of parameters. As these models are being applied more and more to high-stakes scenarios, such as medicine or security (Yang et al., 2019; Gawlikowski et al., 2023), it is critical that their predictions can be trusted. To understand these predictions and their reliability, it is important to differentiate between the concepts of *uncertainty* and *confidence*. According to Lin et al. (2023), uncertainty refers to the 'dispersion' $U(x)$ of the possible predictions for a given input x, while confidence refers to the model's estimated probability of a specific prediction being correct and is expressed in terms of the input and the output as $C(x, y)$. Classical classification models use a `softmax` layer $\sigma(\mathbf{z})_i$ to convert logits $\mathbf{z} = (z_1, z_2, \ldots, z_K)$ into probabilities. The assignment class is usually computed as $\hat{y} = \arg\max_i \sigma(\mathbf{z})_i$ and it is considered the model's

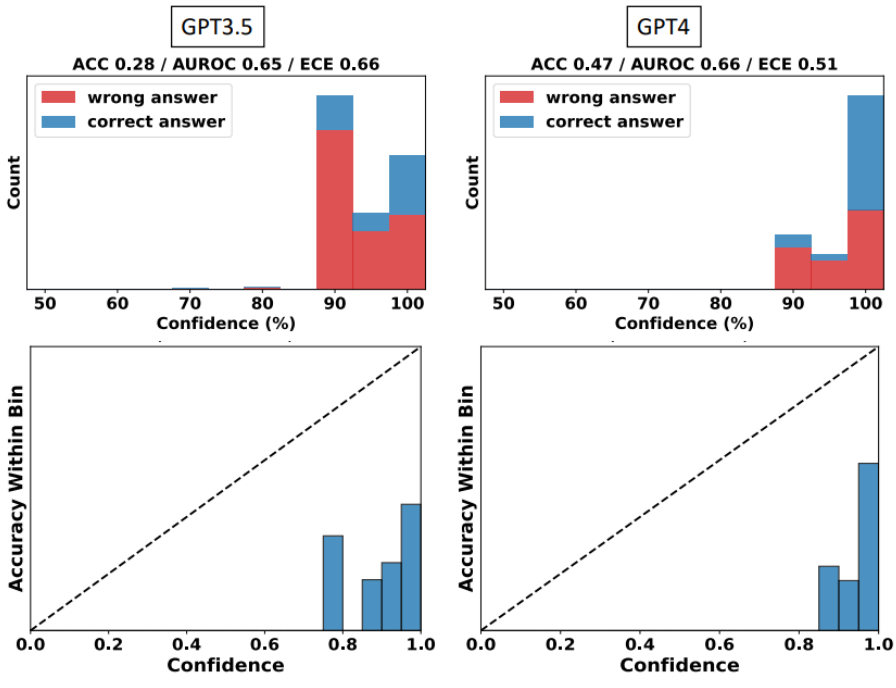


Figure 2.6.1: Empirical distribution (first row) and reliability diagram (second row) obtained by prompting the LLMs to verbalize confidence for a multiple-choice question. The confidence levels are over 80%, while the accuracy in each bin is much lower, indicating overconfidence. Image from Xiong et al. (2023).

prediction (Hendrycks & Gimpel, 2016a). The authors of the study also demonstrate that ignoring the distinction between uncertainty and confidence can lead to incorrect assessments.
Now that the distinction is clear, it is possible to advance to how research tackles the lack of expressed uncertainty. For semantic NLP tasks such as extractive question answering, it is common to use models that predict the start and end tokens of an answer span and provide confidence scores based on the softmax probabilities of these predictions (Devlin et al., 2018; Lan et al., 2019). However, this approach offers no measure to quantify the uncertainty of the prediction. Several works have been proposed in the past years to solve this problem (Xiao et al., 2022; Lin et al., 2023). Common solutions include incorporating uncertainty directly into the model using Bayesian Neural Networks (BNN) (Blundell et al., 2015) or post-hoc methods such as Monte Carlo Dropout (Gal & Ghahramani, 2016), Temperature Scaling (Guo et al., 2017) and Ensemble Learning (Lakshminarayanan et al., 2017).

Bayesian Neural Networks (BNNs) incorporate Bayesian inference principles into neural networks, allowing for the estimation of uncertainty. BNNs treat the weights as distributions, providing a probabilistic interpretation of the network's predictions. If the learning objective is to minimize the negative log likelihood loss $\mathcal{L} = -\frac{1}{N}\sum_i^N \log p(y_i|x_i, w)$, the probability distribution can be computed as shown in Equation 2.6.1, where M represents the set of all possible outcomes.

$$p(y_i = m|x_i, w) = \frac{\exp(f_m(x_i, w))}{\sum_{k \in M} \exp(f_k(x_i, w))}. \tag{2.6.1}$$

For a new input x^* , the predictive distribution is given by Equation 2.6.2.

$$p(y^*|x^*, D) = \int p(y^*|x^*, w)p(w|D)dw. \tag{2.6.2}$$

In practice, due to the high dimensional integral, it is common to approximate Bayesian inference, with methods such as Monte Carlo or Ensembles. In Monte Carlo (MC) Dropout (Gal & Ghahramani, 2016), random sets of weights are set to zero with a certain probability to prevent overfitting. During inference, dropout is applied multiple times to generate a set of stochastic forward passes. Each forward pass corresponds to sampling from the approximate posterior distribution of the network's weights. The predictive mean and variance are estimated by averaging the outputs from these multiple forward passes, as presented in Equations 2.6.3 and 2.6.4 respectively, where T is the number of forward passes, and $f_{\mathbf{w}_t}$ represents the network. While models initialized with a dropout of 0.1 achieve lower uncertainty, Gal & Ghahramani (2016) has shown that during regression tasks, the uncertainty obtained with a value of 0.1 is indistinguishable from 0.2 when the model converges.

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} f_{\mathbf{w}_t}(x^*) \tag{2.6.3}$$

$$\text{Var}(\hat{y}) = \frac{1}{T}\sum_{t=1}^{T} (f_{\mathbf{w}_t}x^* - \hat{y})^2 \tag{2.6.4}$$

Ensemble methods involve training multiple independent models and combining their predictions. These models can be initialized differently or trained on different subsets of the data, using methods such as Bagging, Boosting or Stacking (Breiman, 1996a,b; Freund et al., 1999). In the inference step, the predictions from each model in the ensemble are averaged to produce the final prediction. The mean and variance are computed similarly to Equations 2.6.3 and 2.6.4, where T now represents the number of models in the ensemble. Xiao et al. (2022) found that ensembles used in pretrained LLMs increase the computational cost during finetuning or inference significantly.

Temperature scaling (TS) is a simple yet effective method to scale the softmax predicted probabilities. This is achieved using a temperature parameter τ, which controls the model's confidence by changing the sharpness of the distribution, in a way that stops the predictions from being too overconfident or underconfident (C. Wang, 2023). The calibration of the probabilities is computed according to Equation 2.6.5, and the optimal temperature value is learned by minimizing the negative log-likelihood loss (NLL) on the validation dataset (Equation 2.6.6).

$$p_i = \frac{\exp(g_i/\tau)}{\sum_{j=1}^{k}\exp(g_j/\tau)}, \quad i \in [1 \dots k]. \tag{2.6.5}$$

$$\tau^* = \arg\min_{\tau}\left(-\sum_{i=1}^{N}\log(\text{softmax}(g_i, \tau))\right) \tag{2.6.6}$$

One important goal of uncertainty quantification is to improve *calibration*. According to C. Wang (2023), calibration is defined as the degree of the match between predicted probability p and the true correctness likelihood. Under a classification problem with

an input variable X and a categorical variable $Y \in \{1, 2, ..., k\}$ with k classes, the model f is perfectly calibrated if and only if the condition in Equation 2.6.7 is satisfied.

$$P(Y = y_i | f(X) = p) = p_i \tag{2.6.7}$$

More intuitively, if a model is 90% sure that the sentiment of a document is positive, the predicted outcome should occur 90% of the time in practice. Frequently, even when the prediction probabilities are provided, they are often poorly calibrated (Minderer et al., 2021). This often results in overconfident models (Figure 2.6.1), especially when faced with out-of-distribution data. Some studies attribute this to over-parameterization (C. Wang, 2023), while others suggest that larger pretrained language models are significantly better calibrated in domain and commonsense reasoning (Xiao et al., 2022). The Expected Calibration Error (ECE) is used as a calibration measure, which computes the difference between predicted probabilities ($\text{conf}(B_l)$) and observed frequencies ($\text{acc}(B_l)$) by dividing the prediction space into multiple bins B (Naeini et al., 2015), as shown in Equation 2.6.8. One common way to visualize this metric is using a reliability plot, which shows the confidence of each bin against the accuracy, and reveals specific regions where the model is overconfident or underconfident.

$$\text{ECE} = \sum_{l=1}^{L} \frac{|B_l|}{n} |\text{acc}(B_l) - \text{conf}(B_l)| \tag{2.6.8}$$

To the author's knowledge, when it comes to language models that rely on a visual representation of text, very few studies are dedicated to measuring the levels of uncertainty and confidence in these models. As the field of LLMs rapidly advances, it is paramount that research in explainability and interpretability keeps pace to ensure the effective and transparent deployment of these technologies.

3 Methods

3.1 Data

3.1.1 Pretraining Data

Throughout the majority of the experiments, the model used was the base version of PIXEL, proposed by Rust et al. (2022), which can be found at `https://huggingface.co/Team-PIXEL/pixel-base`. According to the authors, PIXEL was pretrained on the rendered version of the English Wikipedia with 2B and the BookCorpus with 1.1B words (Zhu et al., 2015). The Wikipedia dump used is from the 1st of February 2018, but the exact dataset is not disclosed. However, it is possible to find these dumps at `https://dumps.wikimedia.org/`. Each example in the Wikipedia text dataset contains the content of one full Wikipedia article, after cleaning the markdown and unwanted sections, such as references. Title lines are filtered out since they are short and do not provide enough information.

The BookCorpus dataset was introduced by Zhu et al. (2015) as a way to align books and movies and provide a written explanation for the visual content of a scene. In the context of this study, only the text descriptions are relevant. The dataset contains 17 868 English books of various genres, for instance, Science Fiction, Romance and Adventure. The exact version used to pretrain PIXEL (`https://huggingface.co/datasets/bookcorpusopen`) is no longer available at the moment of writing this thesis, but a similar one can be found in Table 3.1.4.

The original PIXEL model has been pretrained on English data only. Among others, this study proposes a multilingual version, which excludes English. Still, it includes 18 other languages (Amharic, Arabic, Bengali, English, Finnish, Hausa, Igbo, Indonesian, Italian, Kinyarwanda, Korean, Luganda, Naija Pidgin, Norwegian, Romanian, Russian, Swahili, Telugu, Wolof, Yorùbá), in an attempt to determine if lower-resource languages can be successfully used to train a visual language model. This method introduces diversity and speeds up preprocessing, as prerendering is faster on smaller datasets. For this, the Wikipedia dump from the 1st of November 2023 was used. lhttps://huggingface.co/stefania-radu. The prerendered datasets are available at `https://huggingface.co/stefania-radu`. An overview of all languages used to pretrain the multilingual is shown in Table 3.1.1, while the distribution can be observed in Figure 3.1.1. The languages come from 10 language families and 7 scripts, with the most common being Latin.

3.1.2 MasakhaNER 1.0

The dataset used for the Named Entity Recognition (NER) task is MasakhaNER 1.0 (Adelani et al., 2021), a NER benchmark, which includes data from 10 African Languages obtained from local news sources (Amharic, Hausa, Igbo, Kinyarwanda, Luganda, Luo, Nigerian-Pidgin, Swahili, Wolof and Yorùbá), as well as the ConLL-2003 English dataset. The task involves classifying named-entities into nine pre-defined categories, as shown in Table 3.1.2. The MasakhaNER dataset contains labeled entities for each language. The models are trained on the `train` split of the data and tested on the `eval` split.

Language	ISO 639-3	Language Family	Script	Pretraining	Finetuning
Amharic	AMH	Afro-Asiatic	Ge'ez	✓	✓
Arabic	ARA	Afro-Asiatic	Arabic	✓	✓
Bengali	BEN	Indo-European	Bengali	✗	✓
English	ENG	Indo-European	Latin	✗	✓
Finnish	FIN	Uralic	Latin	✓	✓
Hausa	HAU	Afro-Asiatic	Latin	✓	✓
Igbo	IBO	Niger-Congo	Latin	✓	✓
Indonesian	IND	Austronesian	Latin	✓	✓
Italian	ITA	Indo-European	Latin	✓	✗
Kinyarwanda	KIN	Niger-Congo	Latin	✓	✓
Korean	KOR	Koreanic	Korean	✓	✓
Luganda	LUG	Niger-Congo	Latin	✓	✓
Naija Pidgin	PCM	English Creole	Latin	✓	✓
Norwegian	NOR	Indo-European	Latin	✓	✗
Romanian	RON	Indo-European	Latin	✓	✗
Russian	RUS	Indo-European	Cyrillic	✓	✓
Swahili	SWA	Niger-Congo	Latin	✓	✓
Telugu	TEL	Dravidian	Telugu	✓	✓
Wolof	WOL	Niger-Congo	Latin	✓	✓
Yorùbá	YOR	Niger-Congo	Latin	✓	✓

Table 3.1.1: An overview of languages used during pretraining and finetuning. The original PIXEL model is pretrained on English only, while the multilingual version did not use English in the pretraining phase.

3.1.3 GLUE

The Sequence Classification (SC) task relies on the The General Language Understanding Evaluation (GLUE) benchmark (A. Wang et al., 2018). It involves nine sentence-level understanding tasks (CoLA, SST-2, MRPC, QQP, STS-B MNLI-M/MM, QNLI, RTE, WNLI) in English, across three categories: single-sentence tasks, similarity and paraphrase tasks, and inference tasks. As this work focuses on semantic processing in visual language models, the dataset has been chosen due to its wide coverage of semantic phenomena, such as sentence similarity, sentiment classification, and ambiguity. Table 3.1.3 describes the tasks. The models are trained on the `train` split of the data and tested on the `eval` split.

3.1.4 TyDiQA-GoldP

To assess the ability of the model to perform Question Answering (QA), the TyDiQA-GoldP dataset was selected (Clark et al., 2020). It contains nine typologically diverse languages (English, Arabic, Bengali, Finnish, Indonesian, Korean, Russian, Swahili, Telugu) and provides a realistic information seeking task, as the questions are asked by people who are looking for an answer but do not know it yet. The structure of the dataset contains questions written by native speakers, passages with relevant information, and answers provided as short spans of text within the passage. Unlike the primary task, the Gold Passage task focuses more on locating the exact answer within a given context. The models are trained on the `train` split of the data and tested on the `test`

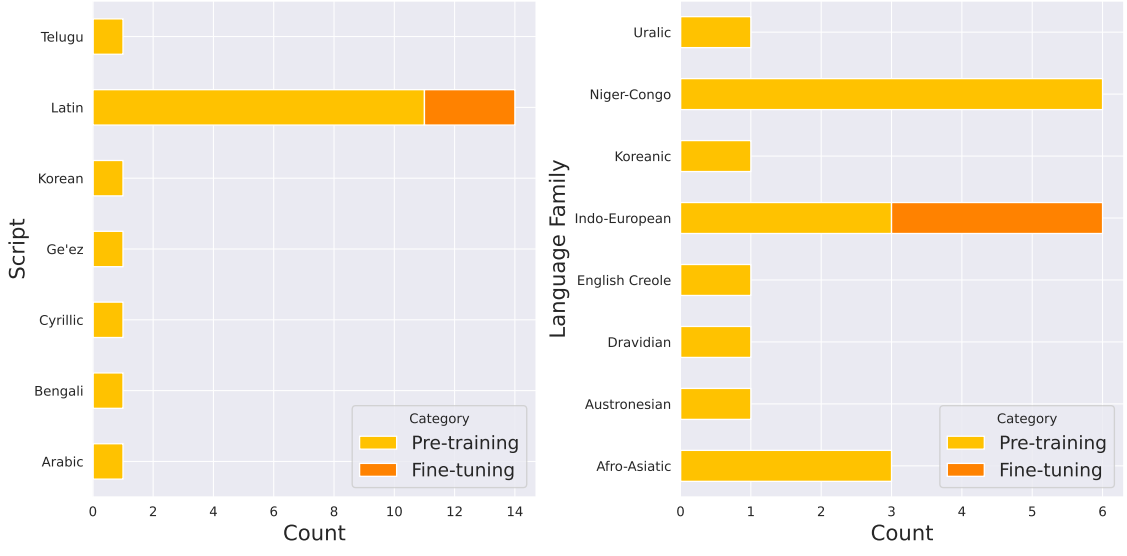


Figure 3.1.1: The distribution of languages used in this study across scripts and language family.

NER Label	Explanation
B-DATE	Beginning of a date entity
B-LOC	Beginning of a location name entity
B-ORG	Beginning of an organization name entity
B-PER	Beginning of a person name entity
I-DATE	Inside a date entity
I-LOC	Inside a location name entity
I-ORG	Inside an organization name entity
I-PER	Inside a person name entity
O	Outside of a named entity

Table 3.1.2: NER Labels with explanations.

split.

3.1.5 Text Renderer

Visual language models require an image-based representation of text, and a very common way to achieve this is through pre-processing via a text renderer. Following the implementation of Rust et al. (2022) and Salesky et al. (2021), The PyGame[1] rendering backend is used to convert the data into 2D images. The original PIXEL model was trained on images generated with PyGame, so for consistency, the same pre-processing step is applied throughout the finetuning experiments which rely on PIXEL, as well as during the zero-shot applications for uncertainty quantification. The renderer supports hieroglyphs scripts, as well as left-to-right and right-to-left writing systems, and text that uses ligatures.

The proposed multilingual model uses the PangoCairo backend, which has also been proposed by Rust et al. (2022). It is based on Pango (Taylor, 2004) and Cairographics

[1] `https://www.pygame.org/`

Name	Description	Task Type	Evaluation Metric
CoLA	Predict if a sentence is grammatically correct or not.	Binary classification	Matthews correlation coefficient (MCC)
SST-2	Predict the sentiment of a sentence (positive/negative).	Binary classification	Accuracy
MRPC	Determine if two sentences are paraphrases.	Binary classification	Accuracy/F1
STS-B	Predict the similarity score between two sentences.	Regression	Pearson/Spearman correlation
QQP	Determine if two questions are paraphrases.	Binary classification	Accuracy/F1
MNLI	Classify the relationship between two sentences (entailment, contradiction, neutral).	Multi-class classification	Accuracy
QNLI	Determine if a sentence answers a question	Binary classification	Accuracy
RTE	Classify relationships between sentences (entailment, not entailment)	Binary classification	Accuracy
WNLI	Determine the referent of a pronoun in a sentence pair	Binary classification	Accuracy

Table 3.1.3: An overview of GLUE tasks

and supports complex text layout, as well as multiple fallback fonts. Consequently, PangoCairo can cover all Unicode codepoints, which is the reason why it is used in the case of the multilingual model that is trained on languages across 7 scripts (see Table 3.1.1).

The text is rendered on a blank RGB image with a height of $H = 16$ pixels, a width of $W = 8464$ pixels, and 3 channels ($C = 3$). Therefore, each image has a resolution of $W \times H \times C = 8464 \times 16 \times 3 = 368 \times 368 \times 3$ pixels. This results in a sequence of 529 patches, each with a size of 16×16 pixels. The color of the text is black and the font size is 8 at 120 DPI. The font family used is the Google Noto Sans fonts collection. Black patches serve as separators and end-of-sequence (`EOS`) markers. Figure 3.1.2 illustrates examples of rendered text from Wikipedia and BookCorpus, using the PyGame renderer. Similarly, Figure 3.1.3 shows examples from Amharic, Arabic, Telugu, and Korean rendered using PangoCairo. Sequences longer than 529 patches are truncated or split into separate sentences.

3.2 Model Architecture

3.2.1 Embeddings

The images generated by the text renderer (Section 3.1.5) are converted into a sequence of non-overlapping patches with a resolution of 16×16 pixels. The sequence is created using three types of embeddings inspired by Dosovitskiy et al. (2020): patch, positional,

Dataset Name	Source
Wikipedia	`https://huggingface.co/datasets/wikimedia/wikipedia`
BookCorpus	`https://huggingface.co/datasets/bookcorpus`
MasakhaNER	`https://github.com/masakhane-io/masakhane-ner`
GLUE	`https://huggingface.co/datasets/nyu-mll/glue`
TyDiQA-GoldP	`https://huggingface.co/datasets/tydiqa`

Table 3.1.4: Dataset used during training and their sources.

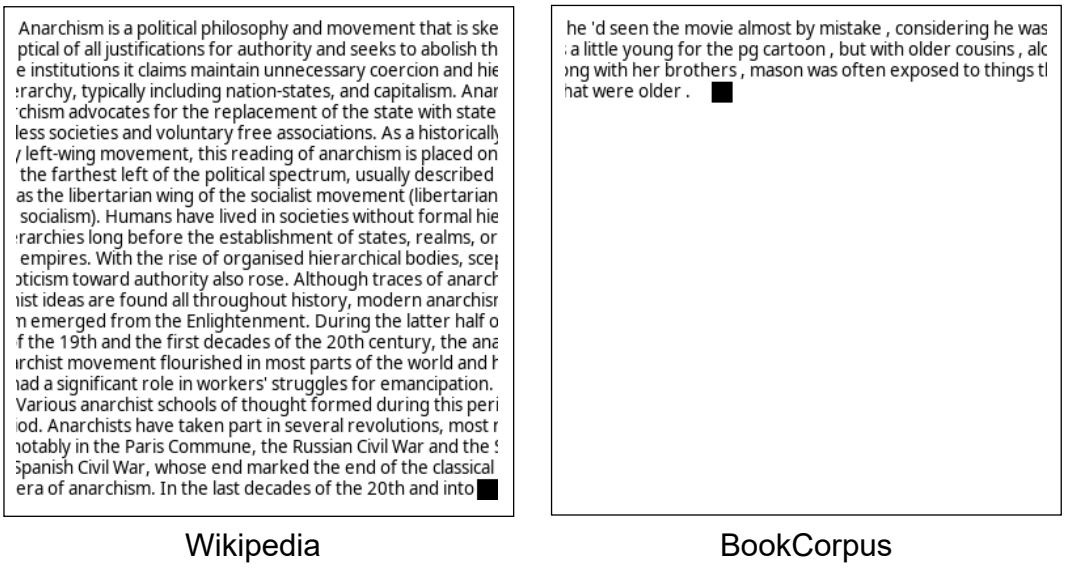


Figure 3.1.2: Examples of English text rendered with the PyGame backend. The black patch represents the `EOS` marker.

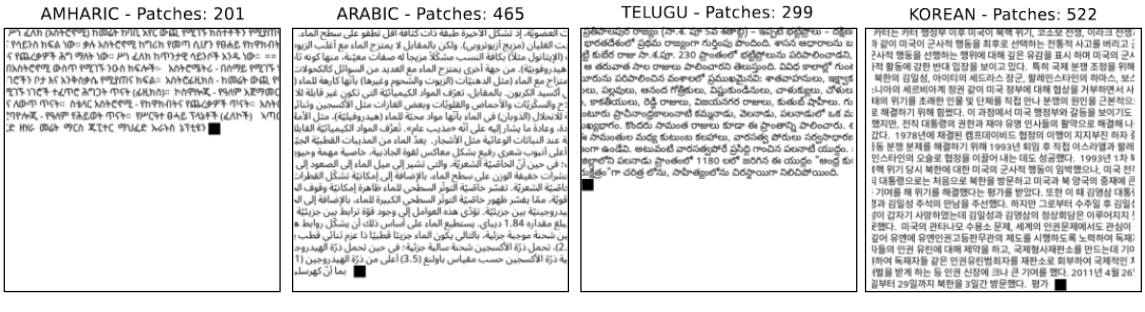


Figure 3.1.3: Examples of text from different languages rendered with the PangoCairo backend (see Table 3.1.1 for more information about the languages). The black patch represents the `EOS` marker.

and CLS embeddings. Refer to Table 3.2.1 for an overview of the parameters used by the model.

Patch Embeddings Each image is represented by a 3D $\texttt{image} \in \mathbb{R}^{H \times W \times C}$, with $H = W = 368$ pixels and $C = 3$ channels. During patch extraction, the image is divided into patches of size $P \times P$ with C channels, where $P = 16$ pixels represent the dimension of a patch. The total (and maximum) number of patches N for an image is calculated as $N = \left(\frac{H}{P}\right) \times \left(\frac{W}{P}\right) = \left(\frac{368}{16}\right) \times \left(\frac{368}{16}\right) = 23 \times 23 = 529$ patches. This leads to a 2D tensor of

image patches represented as $\mathbf{x}_{image} \in \mathbb{R}^{N\times(P^2\times C)}$. Then, to create patch embeddings, each image $\mathbf{x}_{image}$ is projected via a 2D convolutional layer with $C = 3$ input channels and $D_{enc} = (P^2 \times C) = 768$ output channels, representing the encoder hidden size. The kernel and stride size are both equal to the patch size P. The resulting sequence of patch embeddings can be written as $\mathbf{x}_{patch} \in \mathbb{R}^{N\times D_{enc}}$. An overview of this process is presented in Equation 3.2.1.

$$\texttt{image} \in \mathbb{R}^{H\times W\times C} \xrightarrow{\text{Patch Extraction}} \mathbf{x}_{\text{image}} \in \mathbb{R}^{N\times(P^2\times C)} \xrightarrow{\text{2D Convolution}} \mathbf{x}_{\text{patch}} \in \mathbb{R}^{N\times D_{\text{enc}}} \tag{3.2.1}$$

Positional Embeddings Positional embeddings are added to these patch embeddings to encode spatial information in the image. The 2D grid coordinates (x, y) are generated for each patch, and a sine-cosine function is applied to both the x and y coordinates to compute the positional embedding vectors corresponding to the width and the height ($W = H = 23$). These embeddings are concatenated to form a positional embedding vector for each patch. This results in a positional embedding matrix with a size of $(W \times H) \times D_{enc}$. Formally, this can be written as shown in Equations 3.2.2 - 3.2.4.

$$\mathbf{x}_{\text{pos}_{(x,y)}} = [\text{emb}_x, \text{emb}_y] \tag{3.2.2}$$

$$\text{emb}_x = [\sin(x \cdot \omega_i), \cos(x \cdot \omega_i)]_{i=0}^{D_{enc}/2-1} \tag{3.2.3}$$

$$\text{emb}_y = [\sin(y \cdot \omega_i), \cos(y \cdot \omega_i)]_{i=0}^{D_{enc}/2-1} \tag{3.2.4}$$

The frequency ω is defined as $\omega_i = \frac{1}{10000^{2i/D_{enc}}}$ for $i = 0, 1, \ldots, \frac{D_{enc}}{2} - 1$.

CLS Embeddings An additional empty positional embedding is prepended to $\mathbf{x}_{\text{pos}_{(x,y)}}$, as this will contain the classification (CLS) embedding, raising the dimension of embedded patches by one. The CLS token is used for tasks that require the aggregation of information into one prediction, which is the case for word and sentence classification. The resulting final embedding, including the positional information, can be rewritten as $\mathbf{x}_{\text{pos}} \in \mathbb{R}^{(N+1)\times D_{enc}}$.

3.2.2 Span Masking

The model uses a span masking approach in order to mask a sequence of patches in each image, which the decoder has to reconstruct. The algorithm proposed by Rust et al. (2022) and inspired by Raffel et al. (2020) is presented in Algorithm 1. The goal of span masking is to encode more meaningful blocks of text, as opposed to random masking, where each patch has a certain probability of being masked (Figure 3.2.1). This is achieved using the span length parameter S that represents the number of consecutive patches to be masked. Moreover, different sequences are masked with different probability values, in such a way that shorter spans are assigned a larger probability than shorter spans.

Algorithm 1 takes as input a sequence of N patches, the masking ratio R, the span length S, and the cumulative weights C_w. After selecting the span value based on

Parameter	Value
Image size	(16, 8464, 3)
Patch size P	16
Encoder hidden size D_{enc}	768
Encoder intermediate size	3072
Encoder num attention heads	12
Encoder num layers L	12
Decoder hidden size D_{dec}	512
Decoder intermediate size	2048
Decoder num attention heads	16
Decoder num layers K	8
Layer norm ϵ (Ba et al., 2016)	1×10^{-12}
Span masking ratio R	0.25
Span masking max length S	6
Span masking cumulative weights W	{0.2, 0.4, 0.6, 0.8, 0.9, 1}
Span masking spacing	Dynamic
Dropout probability	0.1
Hidden activation	GeLU (Hendrycks & Gimpel, 2016b)
Optimizer	AdamW (Kingma & Ba, 2014)
Adam β	(0.9, 0.999)
Adam ϵ	1×10^{-8}
Weight decay	0.05
Peak learning rate	1.5×10^{-4}
Learning rate schedule	Cosine Decay (Loshchilov & Hutter, 2017)
Minimum learning rate	1×10^{-5}
Learning rate warmup ratio	0.05
Training steps	1M
Batch size	256

Table 3.2.1: The list of parameters used by the model (Rust et al., 2022)

the cumulative weights, the left patch index is randomly chosen from the sequence. Afterwards, the index of the right patch is computed based on the left index and the span, and the resulting masked sequence is added to the set. This continues until the number of masked patches exceeds the allowed number based on the masking ratio. The method ensures that the masked sequences have a dynamic number of unmasked patches in between.

The masking ratio is set to $R = 25\%$ for the base experiments, taking into account the findings of Rust et al. (2022), but the study also investigates the effect of different mask ratios $R_n = 0.1n \quad \text{for} \quad n \in \{1, 2, \ldots, 9\}$ on the reconstruction capabilities of the model. However, given that the model used has been pretrained using a masking ratio of 25%, it is expected that a very large ratio will not yield optimal reconstruction results. Similarly, a set of experiments will be carried out to study the effect of the span length with values $S_n = n \quad \text{for} \quad n \in \{1, 2, \ldots, 6\}$. Throughout these experiments, the maximum sequence length is set to 256 patches, and the spans of $s \in \{1, 2, 3, 4\}$ patches are masked with a probability of 20%, while spans of $s \in \{5, 6\}$ use a probability of 10%. Thus, the vector of cumulative weights is $C_w = (0.2, 0.4, 0.6, 0.8, 0.9, 1)$.

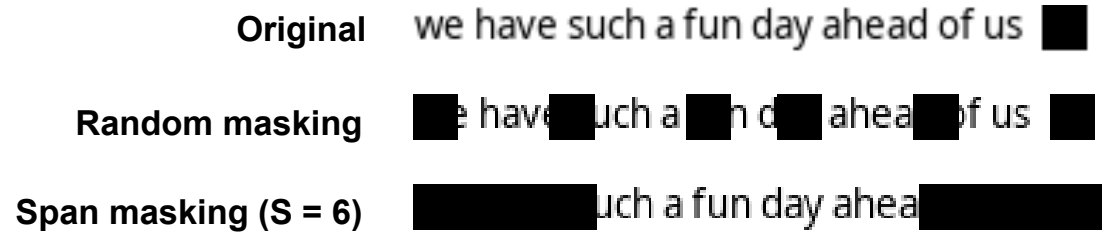


Figure 3.2.1: Example of a sentence masked with random masking and span masking, together with the original. The mask ratio used for both is $R = 25\%$. In the random scenario, the span length is $S = 1$ and each patch is masked with a probability of 50%. In the span masking scenario, the span length is set to $S = 6$, so the maximum number of consecutive masked patches is 6. Spans of $s \in \{1, 2, 3, 4\}$ patches are masked with a probability of 20%, while spans of $s \in \{5, 6\}$ use a probability of 10%

Algorithm 1 Span Masking (Rust et al., 2022)

Input: N, R, S, $C_w = \{w_1, \ldots, w_S\}$
Output: Masked patches $\mathcal{M}$

$\mathcal{M} \leftarrow \emptyset$
repeat
  $s \leftarrow \text{randchoice}(\{1, \ldots, S\}, W)$
  $l \leftarrow \text{randint}(0, \max(0, N - s))$
  $r \leftarrow l + s$
  if $\mathcal{M} \cap \{l - s, \ldots, l - 1\} = \emptyset$ **and** $\mathcal{M} \cap \{r + 1, \ldots, r + s\} = \emptyset$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{l, \ldots, r\}$
  end if
until $|\mathcal{M}| > R$
return $\mathcal{M}$

3.2.3 Encoder

The encoder is based on the VIT-MAE architecture and it processes the visible patches only, which is approximately 396 patches for a masking ratio of 25% and a max sequence length of 529 ($\frac{75}{100} * 529 = 396$). This allows the same encoder to be used for both pre-training and finetuning, since all patches are visible in the finetuning phase. The input to the encoder are the final positional embeddings $\mathbf{x}_{\text{pos}}$ with the concatenated CLS embeddings $\mathbf{c}$. This results in the hidden state $\mathbf{h}_0 = [\mathbf{c}, \mathbf{x}_{\text{pos}}] \in \mathbb{R}^{(1+(1-R)\times N)\times D_{enc}}$. Similarly to the Vaswani et al. (2017) implementation, the encoder contains 12 Transformer layers with Multi-Head Attention, two residual connections, and a Feed Forward block. Each Multi-head Attention block consists of 12 attention heads. An overview of the encoder is shown in Figure 3.2.2.

At the beginning of a Transformer layer, a Normalization layer $\text{Norm}(\mathbf{h}_0) = \frac{\mathbf{h}_0 - \text{E}[\mathbf{h}_0]}{\sqrt{\text{Var}[\mathbf{h}_0] + \epsilon}} * \gamma + \beta$ is applied to the hidden state $\mathbf{h}_0$, where $\gamma = 1e - 5$ is a hyperparameter added for numerical stability, and β is the learnable bias initialized to 0. This is fed to the 12 parallel attention heads in the Multi-Head Attention block, containing the attention function which maps a query and a set of of key-value pairs to an output, computed using the Scaled Dot-Product Attention Vaswani et al. (2017). The outputs of the 12 attention heads are concatenated and projected before applying a second Normalization

layer. Then, a Feed Forward block consisting of a Dense Layer $\text{Linear}(x) = xA^T + b$ maps the hidden size of 768 to the intermediate size of 3072. Finally, a second dense layer followed by Dropout with $p_{\text{hidden}} = 0.1$ is applied to generate the final encoder hidden states $\mathbf{h}_{\text{enc}} \in \mathbb{R}^{(1+(1-R)\times N)\times D_{enc}}$. The Dropout Layer randomly selects a proportion of the weights and sets them to 0, according to $\text{Dropout}(x_i) = \frac{d_i \cdot x_i}{p_{\text{hidden}}}$, where d_i represents a sample from the Bernoulli distribution. In terms of residual connections, one is used to add the embedding's hidden states to the attention outputs, while the second one adds the attention outputs to the final result of the Dropout layer.

3.2.4 Decoder

The decoder is used during pretraining only, to reconstruct the patches of text masked by the span masking algorithm (see Algorithm 1). Firstly, the output of the encoder is projected via a Linear layer to match the decoder hidden size $D_{\text{dec}} = 512$. This results in the decoder embedding $\mathbf{x}_{\text{dec}} \in \mathbb{R}^{(1+(1-R)\times N)\times D_{dec}}$. Given that the decoder does not yet contain the masked patches, the next step is to insert the mask embeddings $\mathbf{x}_{\text{mask}}$ at the corresponding position in the sequence, which can be written as $\left(\mathbf{x}_{\text{dec}} \cup \{\mathbf{x}_{\text{mask}} : i \in \mathcal{M}\}_{i=0}^{N}\right)$. After adding the positional embeddings $\mathbf{x}_{\text{pos}}$, the final decoder hidden states become $\mathbf{d}_0 = \left[\left(\mathbf{x}_{\text{dec}} \cup \{\mathbf{x}_{\text{mask}} : i \in \mathcal{M}\}_{i=0}^{N}\right) + \mathbf{x}_{\text{pos}}\right] \in \mathbb{R}^{(N+1)\times D_{\text{dec}}}$. The decoder consists of 8 Transformer layers with 12 attention heads, which behave the same way as the encoder and generate the output logits $\mathbf{o} \in \mathbb{R}^{D_{\text{dec}}\times(P^2C)}$. The logits represent the reconstructed pixel values for each patch. They are obtained by projecting the outputs of the last Transformer layer using a linear mapping which converts the decoder hidden dimension D_{dec} into patches with the dimension $P^2 \times C$, where P was the patch size and C the number of channels.

3.3 Training

3.3.1 Pretraining

The English PIXEL model is pretrained on a rendered version of the English Wikipedia and the BookCorpus (Zhu et al., 2015). The text is rendered into images $\texttt{image} \in \mathbb{R}^{H\times W\times C}$ using the PyGame renderer and converted into a sequence of patches with size $P = 16$ pixels. The paragraphs are concatenated until the maximum sequence length of $N = 529$ patches is reached. Table 3.2.1 presents an overview of the parameters used during pretraining. The model is trained on a regression-like task to optimize the reconstruction loss which takes the form of a normalized Mean Squared Error (MSE) shown in Equation 3.3.1 The loss is computed only for the set of the masked non-blank patches Q.

$$\mathcal{L} = \frac{1}{|Q|} \sum_{i \in Q} \left|\text{normalize}(x^i_{\text{image}}) - o^i\right|^2 \tag{3.3.1}$$

The multilingual model is pretrained on rendered datasets corresponding to Wikipedia data (`https://huggingface.co/datasets/wikimedia/wikipedia`) from 18 languages (Amharic, Arabic, Finnish, Hausa, Igbo, Indonesian, Italian, Kinyarwanda, Korean, Luganda, Naija Pidgin, Norwegian, Romanian, Russian, Swahili, Telugu, Wolof, Yorùbá), according to Table 3.1.1. The languages were selected to cover a wide variety of scripts and language families. Languages with a very large number of words such as English

have not been included because the resulting rendered images exceed the available disk storage space. The rendered datasets are available publicly on HuggingFace at `https://huggingface.co/stefania-radu`.

3.3.2 Finetuning

The base PIXEL model is finetuned several times throughout the experiments on three tasks: Named-Entity Recognition (NER), Sequence classification (SC), and Question-Answering (QA). The training was completed on Nvidia A100 GPU accelerator cards provided by the Hábrók computing cluster at the University of Groningen, the Netherlands. During ensemble learning, multiple variations of the models are trained on the same data and the predictions are aggregated to combine the knowledge of the different learners. Hyperparameters are chosen to cover a wide range of values, but a parameter grid search was not run due to the very large computational resources required. In the case of tasks such as NER where one word is being rendered at a time, padding is added to create a bijective function between words and patches, so each word appears at the start of an image.

3.4 Uncertainty Quantification

3.4.1 Monte Carlo Uncertainty

The first method used to quantify epistemic uncertainty at the patch level is MC Dropout. Algorithm 2 presents the process of computing patch-level uncertainty in the pretrained base PIXEL model. The input is a rendered $\texttt{image} \in \mathbb{R}^{16\times16\times3}$ with a sequence length of 256 pixels, and the goal is to obtain an uncertainty map $U \in \mathbb{R}^{16\times16\times3}$, containing the uncertainty for each patch. For this, the model is used in 100 forward passes to compute a series of predictions P, which contain per-pixel logits. Then, the mean prediction is created by averaging these logits, resulting in the reconstructed text. A SD image is obtained by computing the SDs of the predictions for each pixel. Since each patch has a dimension of 16×16 pixels, the per-patch uncertainty is defined by averaging the predictions of all SD values inside a patch, and each pixel inside the patch is assigned that value. Finally, the uncertainty map U is a collection of patches representing the overall uncertainty of its pixels. For visualization purposes, the uncertainty map is overlaid on top of the original image, as well as on the reconstructed text. An example is shown at the beginning of this thesis in Figure 1.2.2.

An overall mean uncertainty value is also computed to measure uncertainty at the image level (Equation 3.4.1).

$$\bar{\sigma} = \frac{1}{H \times W} \sum_{h=1}^{H} \sum_{w=1}^{W} \sigma(h, w) \tag{3.4.1}$$

This will be used later on to study the effect of mask ratio and span length on uncertainty in the PIXEL model. Additionally, to examine the relationship between uncertainty and performance, two loss functions are computed during the MC inference: the normalized MSE loss (Equation 3.4.2) used during pretraining and the normalized Gaussian Negative Log-Likelihood (GNLL) loss (Equation 3.4.3), where $eps = 1e-6$ is a clamp value used for stability. Unlike the MSE, the GNLL loss accounts for epistemic uncertainty, by

Algorithm 2 Patch-level Uncertainty with MC Dropout

Input: Rendered image I, model M, # MC samples $N_{\text{MC}} = 100$, dropout rate $p = 0.1$, patch size $P = 16$
Output: Uncertainty map U

Activate dropout in M
for $i \in \{1, \ldots, N\}$ **do**
    $P_i \leftarrow M(I, p)$ ▷ Compute predictions P with dropout
end for
Initialize μ and σ with the shape of I
for each pixel (x, y) **do**
    $\mu(x, y) \leftarrow \frac{1}{N} \sum_{i=1}^{N} P_i(x, y)$
    $\sigma(x, y) \leftarrow \sqrt{\frac{1}{N} \sum_{i=1}^{N} (P_i(x, y) - \mu(x, y))^2}$
end for
Initialize U with the shape of I
for each patch (i, j) in σ **do**
    $\sigma_{\text{patch}} \leftarrow \frac{1}{P^2} \sum_{x=i}^{i+P-1} \sum_{y=j}^{j+P-1} \sigma(x, y)$ ▷ Compute σ per patch
    for $(x, y) \in \{(i, j), \ldots, (i + P - 1, j + P - 1)\}$ **do**
        $U(x, y) \leftarrow \sigma_{\text{patch}}$ ▷ Assign σ_{patch} to all pixels in the patch
    end for
end for
return U

incorporating the variance of the predicted distribution. The predictions are treated as samples from Gaussian distributions with expectations and variances produced by the model. The inputs of the functions are three tensors of the same size: the predictions of the model (reconstructed text), the target image (original text) and a variance image with the per-pixel variance $\text{Var}(x, y) = \frac{1}{N} \sum_{i=1}^{N} (P_i(x, y) - \mu(x, y))^2$.

$$\text{MSE} = \frac{1}{H \times W} (\text{predictions} - \text{image})^2 \tag{3.4.2}$$

$$\text{GNLL} = \frac{1}{2} \left(\log(\max(\text{variance}, \text{eps})) + \frac{(\text{predictions} - \text{image})^2}{\max(\text{variance}, \text{eps})} \right) \tag{3.4.3}$$

3.4.2 Attention Vizualization

Visualizing the attention mechanism in transformer-like models can help us interpret how the model makes its predictions, by showing how different weights are assigned to different parts of the input. For example, attention has been used successfully to detect patterns or biases in language models, such as gender bias (Vig, 2019). However, visualizing attention represents a challenge due to the large number of layers and heads. In the PIXEL model, the encoder contains 12 attention layers with 12 heads, resulting in 144 unique attention structures for each input. Vig (2019) proposed a popular multiscale visualization tool for the text Transformer, that includes a model view and a neuron view of attention. In the case of visual language models, there are no studies up to this date which visualize attention. Here, the difficulty lies in the representation of the input. While, in text models, each attention weight connects two words in the

input, the visual LLM relies on patches consisting of 16×16 pixels. An example is shown in Figure 1.2.3.

To visualize attention in the PIXEL encoder, a square attention grid $A \in \mathbb{R}^{L \times H \times {N_{\text{patches}}}^2}$ is created for the encoded patches, where L is the number of attention layers and H is the number of heads in each layer. This shows model-level attention across all layers and heads for a particular input image. Each cell $A(l,h)$ in this grid visualizes the neuron-level attention weights for a specific head h and layer l. The weights are averaged over 100 Monte Carlo forward passes. Each patch in the original image attends to all other patches, according to the dot product $\frac{\mathbf{q} \cdot \mathbf{k}}{\sqrt{\mathbf{d}}}$ between the query $\mathbf{q}$ of the patch that is paying attention and the key $\mathbf{k}$ of the patch receiving attention (Equation 2.1.2). This results in an attention cell that is a squared image with a dimension of ${N_{\text{patches}}}^2$ patches ($N_{\text{patches}} = 256$). Considering the increased dimensionality of the attention cell, only the first 16 patches are visualized, resulting in an image with 16×16 patches.

3.4.3 Ensemble Learning

Ensemble learning is used to aggregate the predictions of multiple finetuned models. In the context of this thesis, a series of different learners are trained on the same datasets, and a larger model combines their results into one final prediction. For this, a hyperparameter tuning method is used, which involves introducing various values for the batch size, learning rate, and dropout probability. By using multiple configurations, the aim is to improve performance through capturing new patterns and to decrease the variance by combining estimates from different models.

The same overall model architecture is used in solving the QA and NER tasks. This consists of a classification head attached to the PIXEL encoder (Figure 3.2.2). It is made out of a ViT block, a Dropout layer, and a Dense layer. However, given the distinct nature of these tasks, the ensemble learning methods differ in the exact implementation.

QA The QA task involves identifying the correct span (start and end positions) from the context, by predicting the positions within the sequence of tokens where the answer to the question begins and ends. Thus, the task can be reduced to a double multi-class classification over the number of tokens. The loss metric is Cross Entropy (Equation 3.4.4), defined as the average of the individual loss components for the start and end positions. The best logits candidates for each question are selected based on the highest values after applying the `argmax` function. The answer to the question is then the slice between the start and end token. In a regular non-ensemble setting, there is only one finetuned model that dictates the output answer for each example.

$$\mathcal{L}_{\text{QA}} = -\frac{1}{2}\left(\log P_{\text{start}}(y_{\text{start}}) + \log P_{\text{end}}(y_{\text{end}})\right) \tag{3.4.4}$$

In the ensemble learning framework, each example has its own set of candidates that differ between models. There are four main steps to be followed to compute the final prediction for an input question, as presented in Algorithm 3. Each model M_i is applied to the input question q to obtain the candidate answers with corresponding confidence probability values. To reduce the pool of candidates, only the predictions that appear

Algorithm 3 Ensemble QA Prediction

Input: k models $\{M_1, M_2, \ldots, M_k\}$, input question q
Output: Final answer $\hat{a}$ for the question q

$\mathcal{C} \leftarrow \emptyset$
for each model M_i in $\{M_1, M_2, \ldots, M_k\}$ **do**
    $\mathcal{A}_i \leftarrow M_i(q)$ ▷ Get candidate answers and their confidences
    for each candidate a_j in $\mathcal{A}_i$ **do**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{a_j\}$
    end for
end for
$\mathcal{C} \leftarrow \left\{c \mid \sum_{i=1}^{k} \mathbf{1}_{c \in \mathcal{A}_i} = k\right\}$ ▷ Keep the candidates that appear in all models
for each candidate c in $\mathcal{C}$ **do**
    $\text{conf}_c \leftarrow \frac{1}{k} \sum_{i=1}^{k} \text{confidence}_{M_i}(c)$ ▷ Compute average confidence
end for
$\hat{a} \leftarrow \arg\max_{c \in \mathcal{C}} \text{conf}_c$ ▷ Select candidate with highest confidence
return $\hat{a}$

in all models are kept. The average confidence conf_c is computed for each candidate across all models. Finally, the candidate with the highest confidence is selected.

NER The NER task involves assigning a label to each token from a list of 9 predefined classes (see Table 3.1.2). A series of models with different hyperparameter configurations are finetuned on the data. Their predicted logits are averaged and combined into one value for each class. The final label is computed as shown in Equation 3.4.5, where L is the set of labels (classes) and k is the number of models. The loss metric is Cross Entropy (Equation 3.4.6), defined based on the number of labels.

$$\text{label} = \arg\max_{l \in L} \left(\frac{1}{k} \sum_{i=1}^{k} \text{logits}_{i,l} \right) \tag{3.4.5}$$

$$\mathcal{L}_{\text{NER}} = -\sum_{l \in L} y_l \log P_l \tag{3.4.6}$$

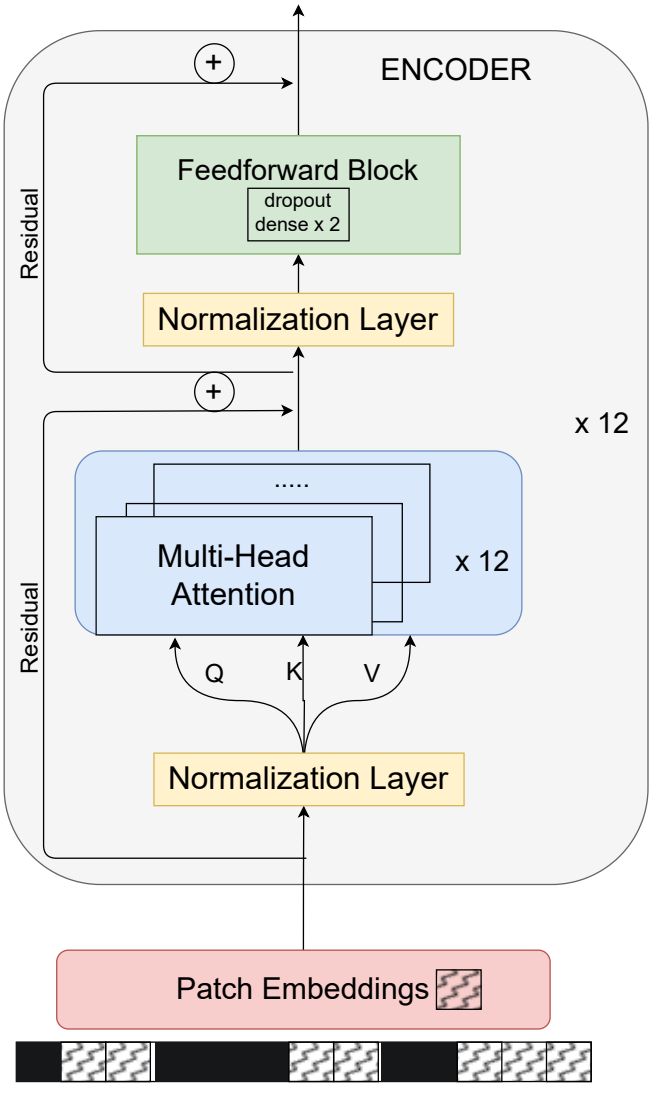


Figure 3.2.2: The encoder of the visual language model. Unlike the Transformer encoder (Vaswani et al., 2017), the current encoder applies the normalization layer before and after attention. The input to the encoder is a sequence of patch embeddings corresponding to the visible patches.

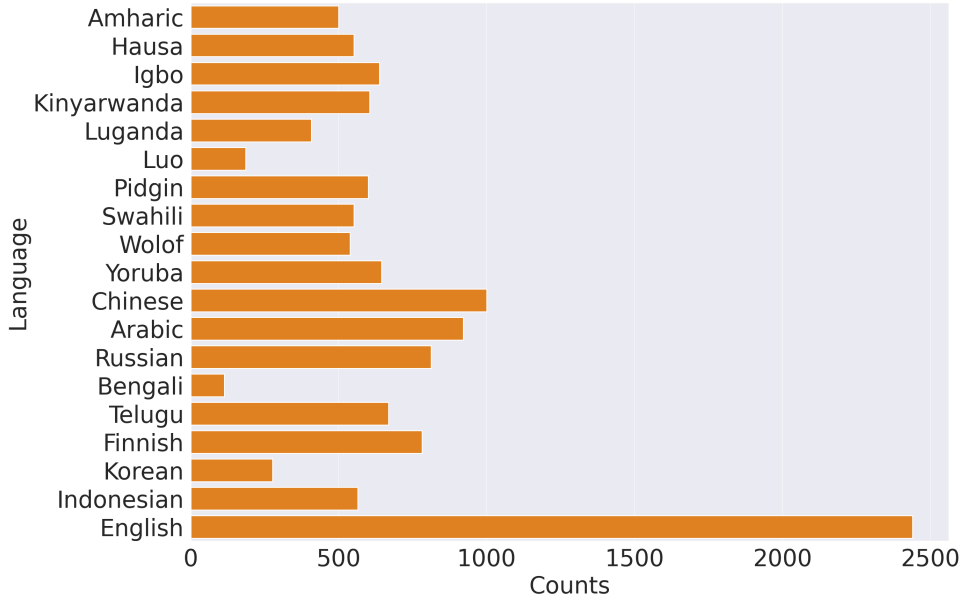


Figure 4.1.1: Distribution of languages used in the MC uncertainty experiments.

4 Experimental Setup

4.1 Monte Carlo Uncertainty

Data from 19 languages and 3 datasets is used to complete the experiments in this section: MasakhaNER 1.0 (Section 3.1.2), GLUE (Section 3.1.3) (only 1000 datapoints from the COLA subset are randomly selected) and TyDiQA-GoldP (Section 3.1.4). The number of samples for each language is shown in Figure 4.1.1. The English samples come from all three datasets to ensure more semantic diversity and variety in terms of text length.

The overall MC uncertainty is defined as the mean uncertainty (Equation 3.4.1) across all images in a specific category: task, language, or script. It is presented in terms of performance metrics and model parameters. The performance metrics include the MSE loss (Equation 3.4.2), used to pretrain the PIXEL model, and the GNLL loss (Equation 3.4.3), which incorporates uncertainty in the loss measure. The model was already trained with a dropout rate of $p = 0.1$, meaning that 10% of the weights are set to zero. The number of forward passes or MC samples is $N_{MC} = 100$, which is enough to get an accurate estimate of the mean and standard deviation (SD). Given that there is no scientific consensus about the perfect masking ratio in language models (Wettig et al., 2022) and that visual language models are still in their infancy, this experiment will also study the effect on uncertainty of different masking ratios, as well as different span lengths. An overview of the experiments in this section is presented in Table 4.1.1.

4.1.1 Uncertainty Across Tasks

The first set of experiments looks at uncertainty across tasks: NER (MasakhaNER 1.0), SC (GLUE), and QA (TyDiQA-GoldP). In NER, the model is required to understand word-level information and remember long-term dependencies between words. The SC task tests sentence-level understanding across a wide range of categories, such as sentence

Experiment	Data	Hyperparameters	Metrics
MCU Tasks (Section 4.1.1)	NER (MasakhaNER 1.0), SC (GLUE), QA (TyDiQA-GoldP)	$R \in \{0.1, 0.2, \ldots, 0.9\}$, $S \in \{1, 2, \ldots, 6\}$, $W = \{0, 0, \ldots, 0, 1\}, \lvert W\rvert = \lvert S\rvert$	MSE GNLL Uncertainty ($\bar{\sigma}$)
MCU Scripts (Section 4.1.2)	Latin, Ge'ez, Chinese Characters, Arabic, Cyrillic, Bengali, Telugu, Korean	$R \in \{0.1, 0.2, \ldots, 0.9\}$, $S \in \{1, 2, \ldots, 6\}$, $W = \{0, 0, \ldots, 0, 1\}, \lvert W\rvert = \lvert S\rvert$	MSE GNLL Uncertainty ($\bar{\sigma}$)
MCU Languages (Section 4.1.3)	Amharic, English, Hausa, Igbo, Kinyarwanda, Luganda, Luo, Nigerian Pidgin, Swahili, Wolof, Yoruba, Chinese, Arabic, Russian, Bengali, Telugu, Finnish, Korean, Indonesian	$R \in \{0.1, 0.2, \ldots, 0.9\}$, $S \in \{1, 2, \ldots, 6\}$, $W = \{0, 0, \ldots, 0, 1\}, \lvert W\rvert = \lvert S\rvert$	MSE GNLL Uncertainty ($\bar{\sigma}$)
VU (Section 4.1.4)	Nigerian Pidgin, Igbo	$R = 0.25$, $S = 6$, $W = \{0.2, 0.4, 0.6, 0.8, 0.9, 1\}$	GNLL Uncertainty ($\bar{\sigma}$)
CA (Section 4.1.5)	NER (MasakhaNER 1.0), SC (GLUE), QA (TyDiQA-GoldP)	$R = 0.25$, $S = 6$, $W = \{0.2, 0.4, 0.6, 0.8, 0.9, 1\}$	RMSE Uncertainty ($\bar{\sigma}$)

Table 4.1.1: Overview of the MC Uncertainty experiments. MCU = Monte Carlo Uncertainty; VU = Visualizing Uncertainty; CA = Calibration Analysis.

similarity or inference. Being able to answer questions and extract relevant information from text is what makes the QA task anchored in real-life, but also challenging. As all these tasks measure semantic processing at different levels, they provide a reliable measure of the overall capacity of the model to process uncertainty.

4.1.2 Uncertainty Across Scripts

Studying uncertainty with respect to different scripts is necessary to build reliable models that can scale up to real-world applications where many scripts are often encountered. One of the advantages of understanding the relationship between scripts and uncertainty is that it can help researchers allocate resources more fairly and increase the amount of data gathered for less-represented scripts. This experiment will look at MC uncertainty across 8 scripts: Latin, Ge'ez, Chinese Characters, Arabic, Cyrillic, Bengali, Telugu, and Korean. As mentioned previously, the effect of mask ratio and span length is also analysed.

4.1.3 Uncertainty Across Languages

To further increase the granularity of the analysis, another experiment focuses on uncertainty across languages. Similar to scripts, languages have varying complexities and characteristics, and evaluating uncertainty can help with bias detection and mitigation, as well as with understanding the causes behind performance imbalances. There are 19 languages used throughout this experiment: Amharic, English, Hausa, Igbo, Kinyarwanda, Luganda, Luo, Nigerian Pidgin, Swahili, Wolof, Yoruba, Chinese, Arabic,

Russian, Bengali, Telugu, Finnish, Korean, Indonesian. Following the structure of previous experiments, the hyperparameters mask ratio and span length are also included in this study.

4.1.4 Visualizing Uncertainty in Text Reconstruction

Apart from a high-level view of uncertainty across different categories, this experiment proposes a more detailed outlook and aims to visualize MC uncertainty at the patch level. For this, the top and bottom 5 performances with respect to the GNLL loss have been selected. Moreover, the experiment includes an example outside of the dataset consisting of text written by the author for the proposal of this thesis (Figure 1.2.2).

4.1.5 Calibration Analysis

This experiment investigates the calibration of visual language models by examining the relationship between model performance and uncertainty across tasks and languages. The same set of tasks and languages as in experiments 4.1.2 and 4.1.3 are used. The performance of the models is measured using Root Mean Square Error (RMSE = $\sqrt{\text{MSE}}$, Equation 3.4.2), while uncertainty is quantified using MC standard deviation. By analyzing this data, the goal is to evaluate how well the predicted uncertainties align with actual performance errors across the different scripts and languages. The findings will cast light on the robustness of visual language models in multilingual and multi-task settings, highlighting areas where predictions are more uncertain and potentially less accurate.

4.2 Attention Vizualization

In this experiment, the goal is to visualize the attention weights in the PIXEL model as an attention grid. The data was selected based on the findings from the Monte Carlo experiments (Section 4.1.1). To study if there is any difference in the levels of attention and the patches that the model is paying attention to, the experiment compares the input image with the lowest MC uncertainty (Nigerian Pidgin) to that with the highest uncertainty (Igbo) across all datapoints from the test split of the three datasets: MasakhaNER 1.0 (Section 3.1.2), GLUE (Section 3.1.3) and TyDiQA-GoldP (Section 3.1.4). A third example was randomly selected as well for comparison.

4.3 Ensemble Learning

4.3.1 Extractive Question Answering

There are 4 learner models finetuned on each of the 9 languages of the TyDiQA-GoldP (Section 3.1.4) dataset to perform extractive question-answering, resulting in 36 total models. Each model is a variation of the PIXEL base, which had been pretrained on the English Wikipedia and the BookCorpus datasets, according to Table 3.2.1. The 4 finetuning configurations can be seen in Table 4.3.1. Each model is trained on the `train` split of a language in the dataset and evaluated on the `validation` split of the same language. All models use a sequence length of 128 patches, meaning that all questions that render to more than 128 patches will be truncated. For each input question, one model outputs 20 candidate answers. The maximum number of optimization steps is

20 000, and the optimization measure is the F1 score. The F1 score is defined in terms of the True Positive tokens (tokens correctly predicted as part of the answer), False Positive tokens (tokens incorrectly predicted as part of the answer), and False Negative tokens (tokens that are part of the true answer but not predicted). Only the values of the batch size (BSZ), learning rate (LR), dropout probability (DP), and the seed are changed as presented in Table 4.3.1.

4.3.2 Named Entity Recognition

There are 5 learner models finetuned on each of the 10 languages of the MasakhaNER 1.0 dataset (Section 3.1.2) dataset to perform Named Entity Recognition, resulting in 50 total models. Similar to the QA setup, each model is a variation of the PIXEL base, which had been pretrained on the English Wikipedia and the BookCorpus datasets, according to Table 3.2.1. The 5 finetuning configurations can be seen in Table 4.3.2. Each model is trained on the `train` split of a language in the dataset and evaluated on the `test` split of the same language. All models use a sequence length of 196 patches and each token is renderer at the beginning of the image. The maximum number of optimization steps is 15 000, and the optimization measure is the F1 score. Only the values of the batch size (BSZ), learning rate (LR), dropout probability (DP), and the seed are changed as presented in Table 4.3.2.

Parameter	Value
Common Parameters	
Dataset name	tydiqa
Dataset config name	secondary_task
Sequence length	400
Stride	160
Question max length	128
Gradient accumulation steps	1
Max steps	20000
Number of train epochs	10
Early stopping	True
Early stopping patience	5
Evaluation metric	$F1 = \frac{2\times TP}{2\times TP+FP+FN}$
Doc stride	160
Number of best predictions	20
Model 1	
Batch size	32
Learning rate	7×10^{-4}
Dropout probability	0.15
Seed	101
Model 2	
Batch size	16
Learning rate	7×10^{-5}
Dropout probability	0.15
Seed	102
Model 3	
Batch size	8
Learning rate	7×10^{-5}
Dropout probability	0.05
Seed	103
Model 4	
Batch size	32
Learning rate	7×10^{-6}
Dropout probability	0.1
Seed	104

Table 4.3.1: The finetuning configuration of the QA models, including the common parameters and those changed among the 4 learners.

Parameter	Value
Common Parameters	
Dataset name	masakhane-ner
Sequence length	196
Gradient accumulation steps	1
Max steps	15000
Number of train epochs	10
Early stopping	True
Early stopping patience	5
Evaluation metric	$F1 = \frac{2\times TP}{2\times TP+FP+FN}$
Model 1	
Batch size	64
Learning rate	5×10^{-5}
Dropout probability	0.1
Seed	100
Model 2	
Batch size	64
Learning rate	5×10^{-6}
Dropout probability	0.2
Seed	101
Model 3	
Batch size	32
Learning rate	5×10^{-5}
Dropout probability	0.1
Seed	102
Model 4	
Batch size	32
Learning rate	5×10^{-6}
Dropout probability	0.1
Seed	103
Model 5	
Batch size	16
Learning rate	5×10^{-5}
Dropout probability	0.2
Seed	104

Table 4.3.2: The finetuning configuration of the NER models, including the common parameters and those changed among the 5 learners.

5 Results

5.1 Monte Carlo Uncertainty

5.1.1 Uncertainty Across Tasks

Mask Ratio The MC uncertainty as defined in Section 4.1 is first presented across tasks in Figures 5.1.1 – 5.1.6. To study the effect of mask ratio, two loss metrics (MSE and GNLL) and the overall uncertainty are plotted against the different mask ratio values. Each line represents a task, corresponding to one of the three datasets. The general trends can be observed in Figures 5.1.1, 5.1.2, and 5.1.3.

The results from Figure 5.1.1 indicate that the loss increases with the mask ratio. This is expected as the model was trained to reconstruct the image patches with a mask ratio of $R = 0.25$ (see the pretraining configuration in Table 3.2.1). There is also a wide performance gap between the sequence classification task (GLUE) and the rest of the tasks, which can be attributed to language. The GLUE dataset contains English text, the language the PIXEL model was pretrained on, while TyDiQA-GoldP and MasakhaNER are multilingual datasets. In terms of the GNLL loss (Figure 5.1.2), which combines the performance component with uncertainty, GLUE remains associated with the lowest loss across all R values. On the other hand, the GNLL loss is highest for the TyDiQA-GoldP dataset. However, when studying the MC uncertainty results in Figure 5.1.3, GLUE achieves the highest overall uncertainty, which indicates that pixel-level uncertainty increases with text that has more semantic complexity, as it is the case in sentiment classification, semantic similarity or textual entailment tasks.

Apart from the mean values, the distribution of the results is also shown in Figures 5.1.4, 5.1.5, and 5.1.6. When it comes to the pretraining loss (Figure 5.1.4), the spread of the distribution is generally broad, but it decreases as the mask ratio increases. The MC uncertainty (Figure 5.1.6) and GNLL loss (Figure 5.1.5) plots suggest that lower mask ratios (0.1 to 0.3) generally correspond to lower uncertainty across all datasets, indicating that less masking leads to more certain predictions. In this case, the largest part of the data is concentrated between uncertainty values of 0.15 and 0.25. As the mask ratio increases, the distribution becomes more spread out.

Span Length The span length measures the number of consecutive masked patches the model has to reconstruct. In other words, it can be interpreted as the ability of the model to deal with larger contexts and predict longer segments of text. Across datasets (Figures 5.1.7, 5.1.8, and 5.1.9), the performance decreases with larger span length values ($S > 2$), especially for the GLUE dataset. This indicates that the model is struggling to reconstruct multiple patches at the same time, which can be attributed to a lack of contextual understanding, partly due to the unigram patch embeddings used. In the case of MasakhaNER and TyDiQA-GoldP, the level of variation in predictions stays constant across all span length values, with a slight increase for $S > 4$. The distributions in Figures 5.1.10, 5.1.11, and 5.1.12 show that there are two peaks within each dataset, which can represent the difficult and easy examples. This is more pronounced in the case of the NER task (Figure 5.1.12), where a high proportion of the words fall under the non-entity (O) class, and less pronounced in GLUE, which contains a multitude of subtasks.

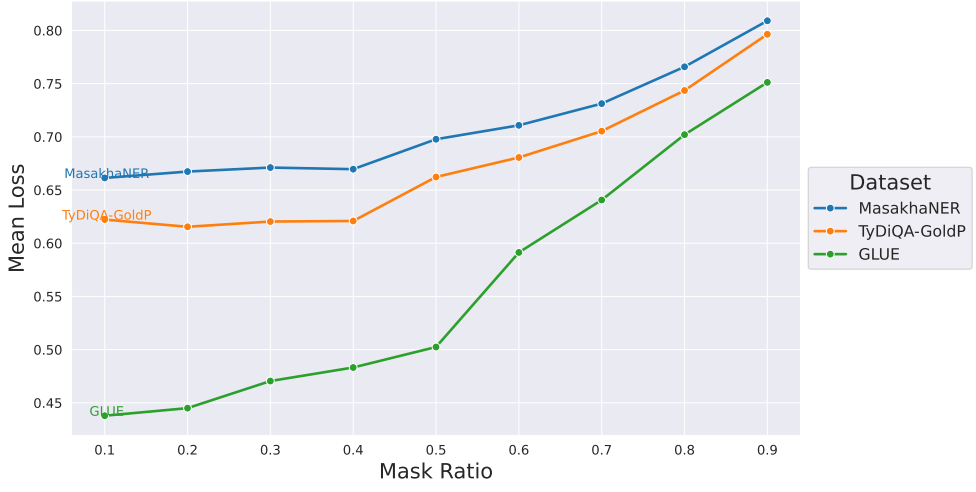


Figure 5.1.1: Mean MSE Loss across the different datasets for each mask ratio value R.

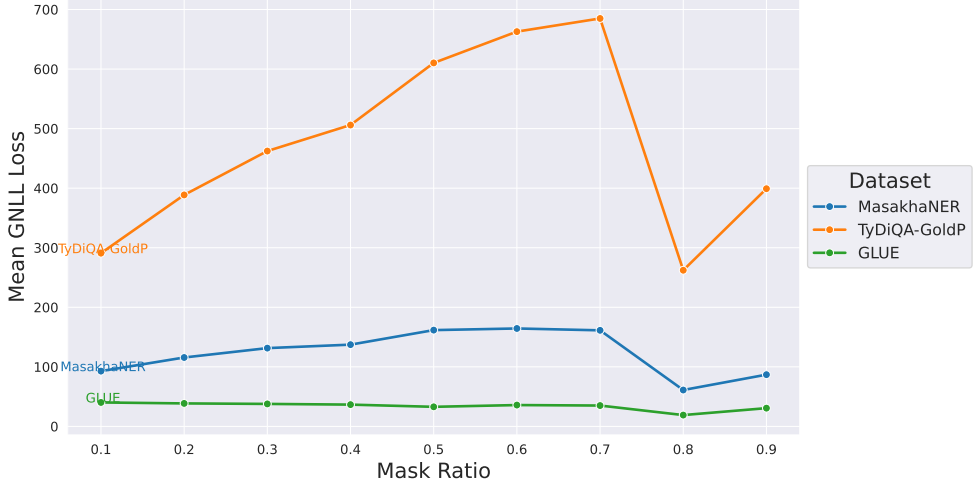


Figure 5.1.2: Mean GNLL Loss across the different datasets for each mask ratio value R.

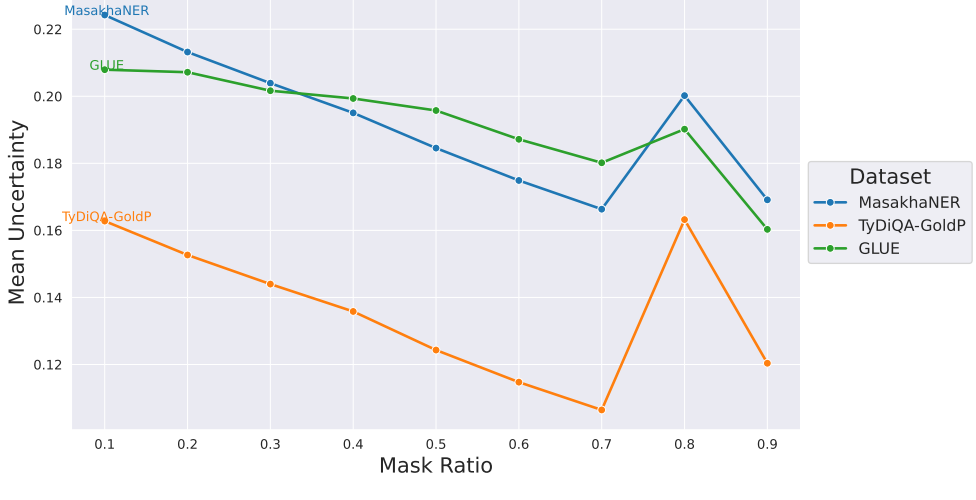


Figure 5.1.3: Mean MC Uncertainty across the different datasets for each mask ratio R.

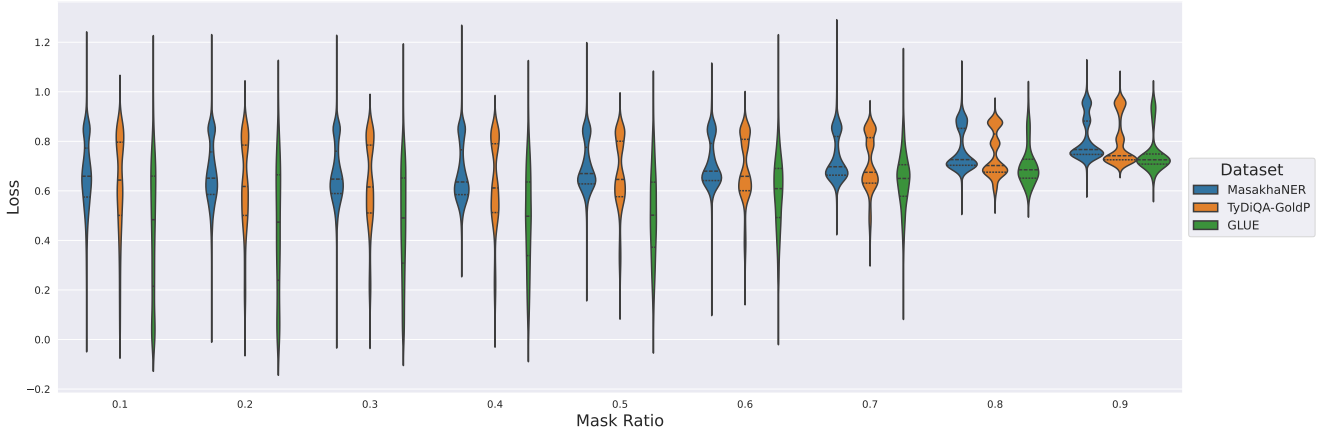


Figure 5.1.4: The distribution of the MSE Loss across the different datasets for each mask ratio R.

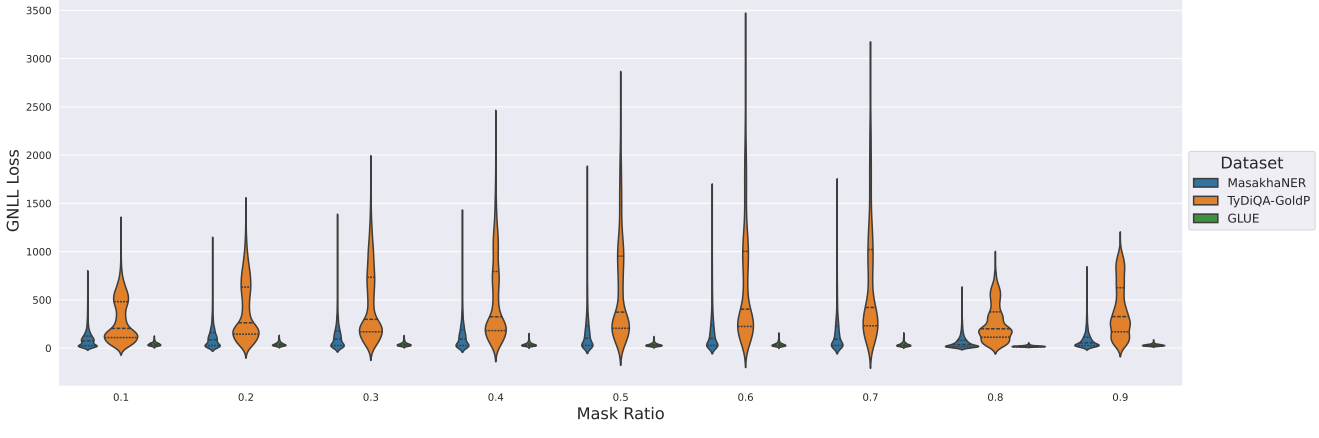


Figure 5.1.5: The distribution of the GNLL Loss across the different datasets for each mask ratio R.

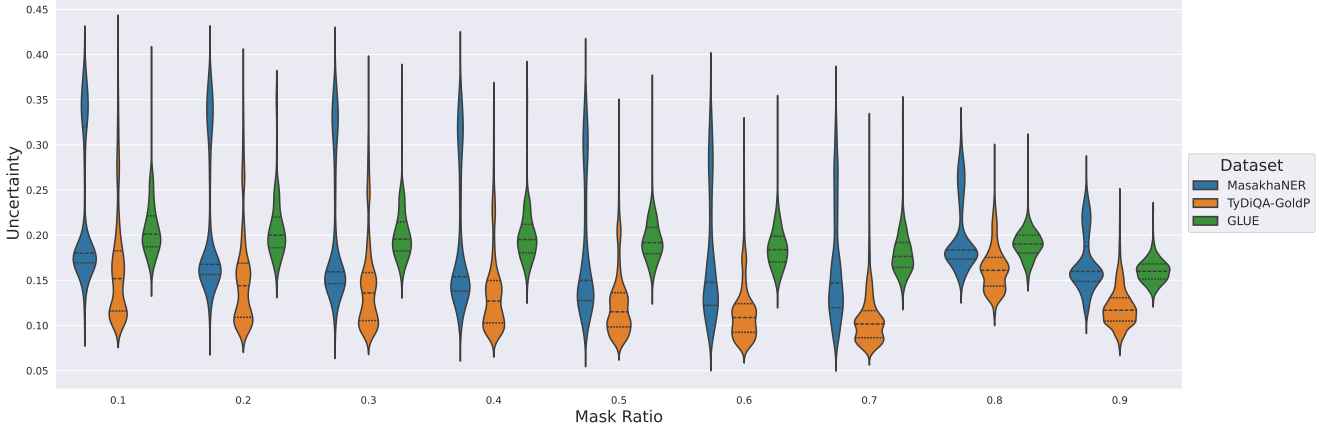


Figure 5.1.6: The distribution of the MC Uncertainty across the different datasets for each mask ratio value R.

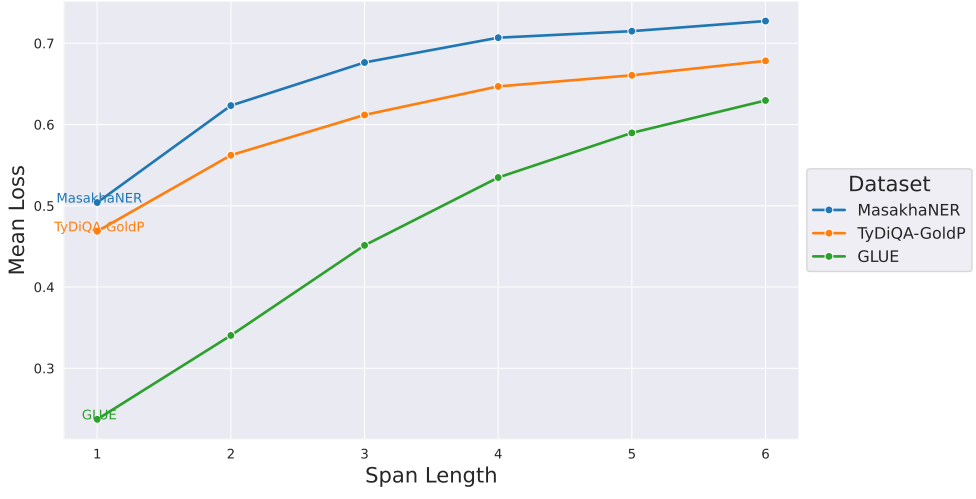


Figure 5.1.7: Mean MSE Loss across the different datasets for each span length S.

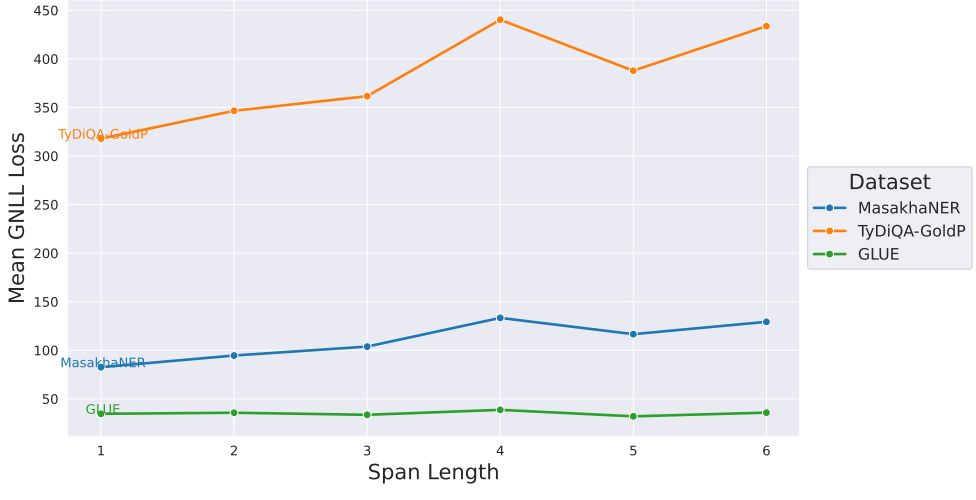


Figure 5.1.8: Mean GNLL Loss across the different datasets for each span length S.

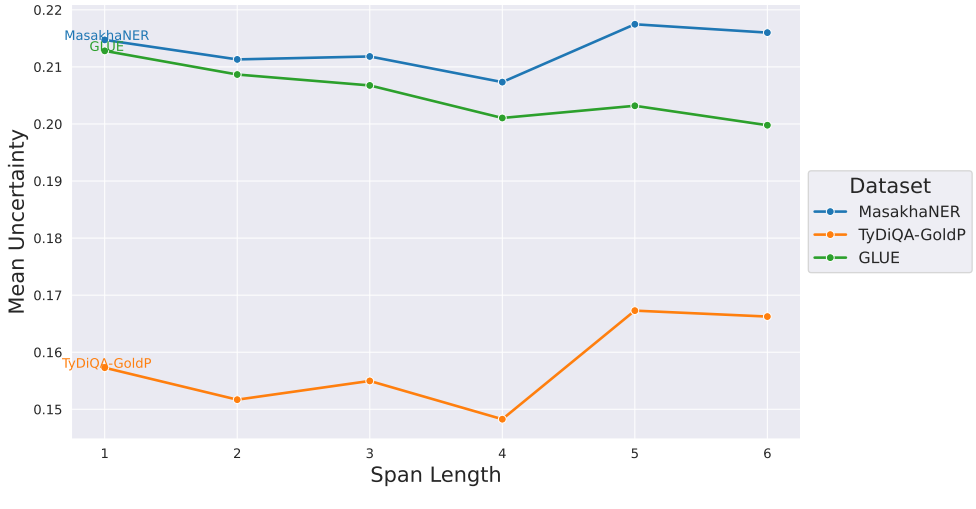


Figure 5.1.9: Mean MC Uncertainty across the different datasets for each span length S.

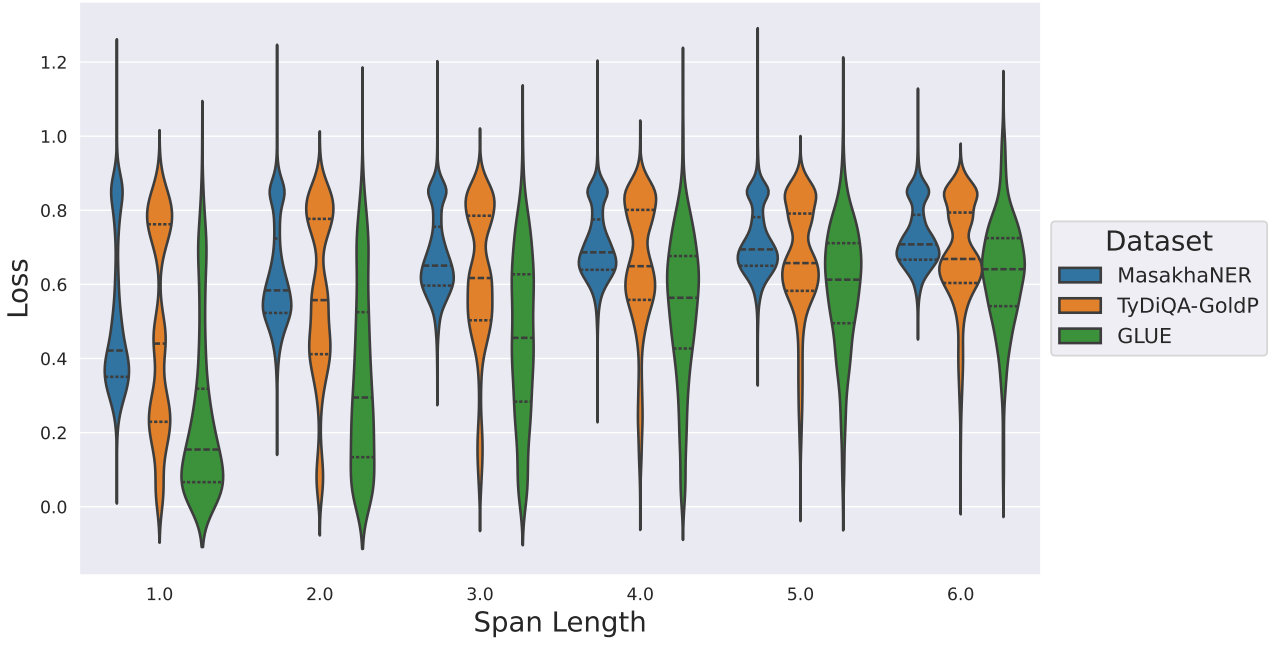


Figure 5.1.10: The distribution of the MSE Loss across the different datasets for each span length S.

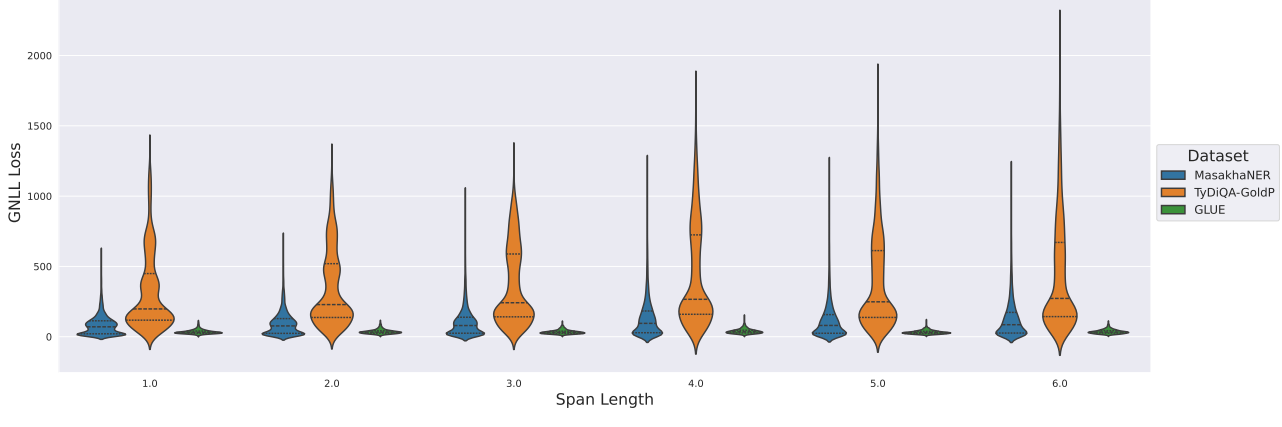


Figure 5.1.11: The distribution of the GNLL Loss across the different datasets for each span length S.

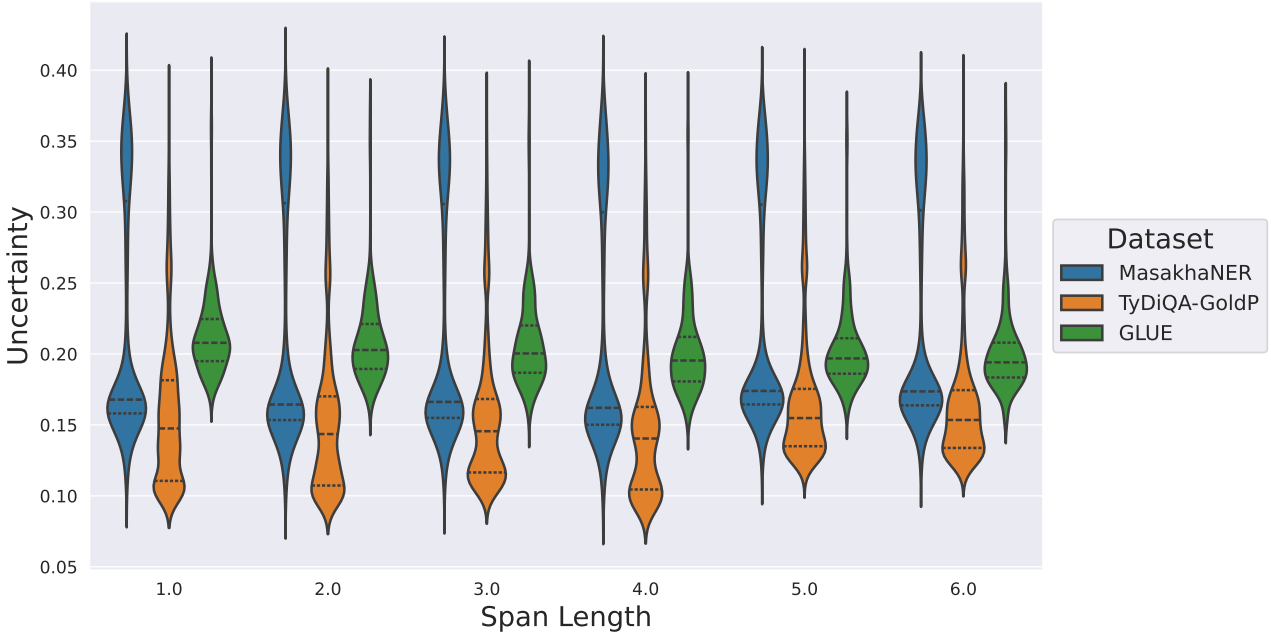


Figure 5.1.12: The distribution of the MC Uncertainty across the different datasets for each span length S.

5.1.2 Uncertainty Across Scripts

Mask Ratio A similar analysis to the one proposed in the previous section is carried out concerning the different scripts from the datasets. The findings are presented in Figures 5.1.13 – 5.1.6. The overall trends show that Ge'ez, Chinese Characters, Arabic, and Korean scripts exhibit high mean loss values and the increase is more pronounced at mask ratios above 0.6. The Latin and Cyrillic scripts are increasing more gradually with a sharper uptick around 0.8 – 0.9. The lowest MSE loss and uncertainty are associated with the Latin and Cyrillic scripts. The main script found in the English Wikipedia and the BookCorpus is Latin, and there is a high overlap between Latin and Cyrillic characters, given that both scripts share Greek as a common ancestor. However, the uncertainty in the Cyrillic script is lower, compared to Latin, while the loss is higher in Cyrillic, suggesting that there is a lack of calibration inside the model. The scripts with the highest MC uncertainty (Figure 5.1.15) are Ge'ez and Chinese Characters, both of which are visually quite distinct from the Latin script.

In terms of the distributions, Figures 5.1.16, 5.1.17, and 5.1.18 show that there is a large variation in performance, especially for the Latin script (Figures 5.1.16 and 5.1.17). This demonstrates that some examples are significantly more visually challenging than others even within a script. The variation decreases with the increase in mask ratio, as does the loss. The variations in uncertainty (Figure 5.1.18) appear to remain relatively constant throughout the different mask ratio values. However, there seems to be a slight decrease in Ge'ez and Chinese Characters around the extreme endpoints of the mask ratio interval, around 0.1 – 0.2 and 0.8 – 0.9. When looking at the GNLL loss (Figure 5.1.17), Latin has a large number of outliers, when compared to Arabic or Telugu.

Span Length Figures 5.1.19, 5.1.20, and 5.1.21 show little variation in performance and uncertainty when analyzing the 8 different scripts across different span length val-

ues. In terms of loss, there is an increase in MSE (Figure 5.1.19) and GNLL (Figure 5.1.20) with span length for the Latin and Cyrillic scripts. The rest of the scripts remain constant for all span length values. Moreover, there is little to no change in MC uncertainty (Figure 5.1.21) across span length, which indicates that masking larger text segments does not reduce the reliability of the predictions. The findings in Figures 5.1.22, 5.1.23, and 5.1.24 do not present any significant variations in the distribution of the results for the different scripts.

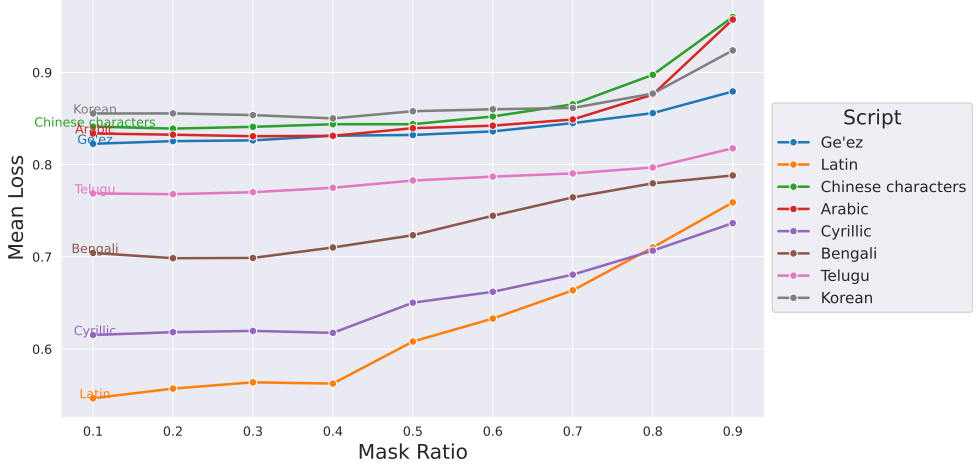


Figure 5.1.13: Mean MSE Loss across the scripts for each mask ratio value R.

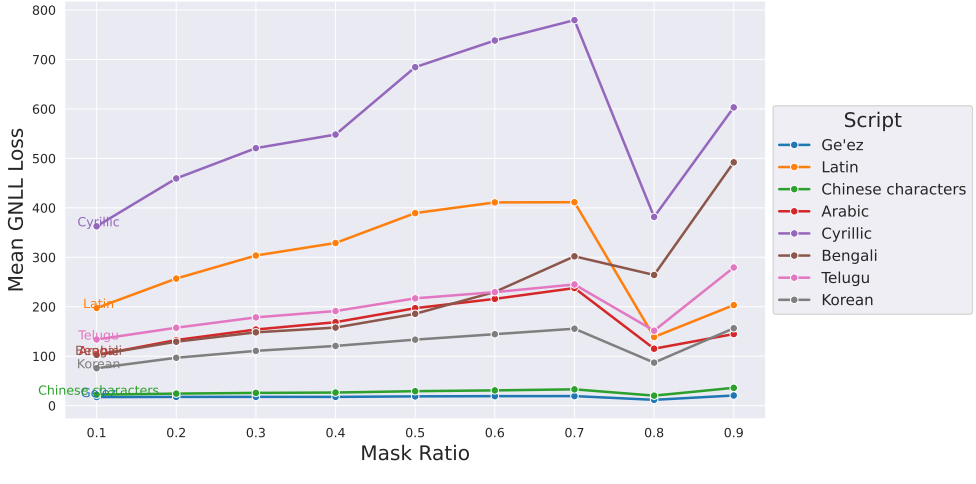


Figure 5.1.14: Mean GNLL Loss across the scripts for each mask ratio value R.

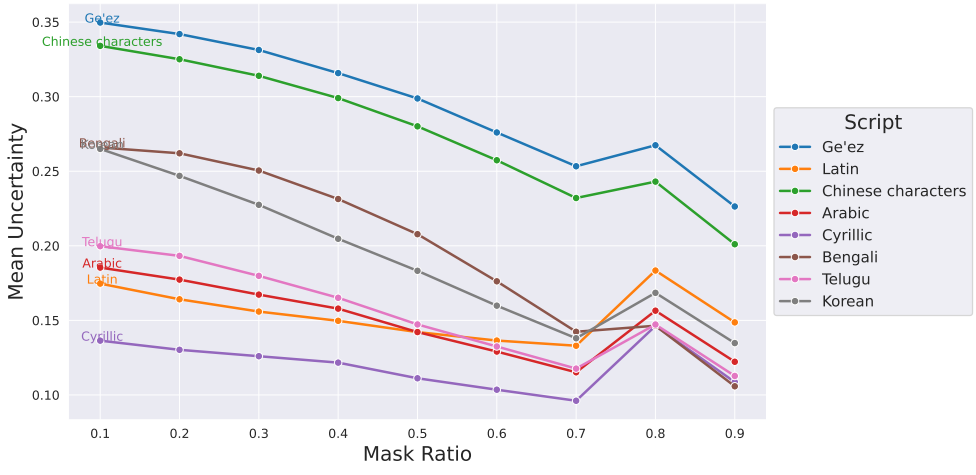


Figure 5.1.15: Mean MC Uncertainty across the scripts for each mask ratio value R.

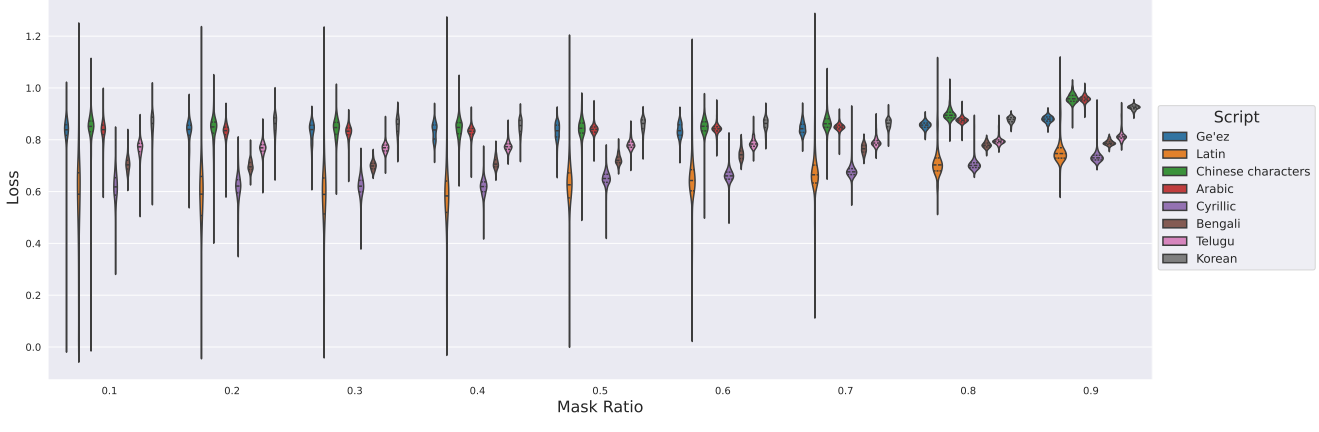


Figure 5.1.16: The distribution of the MSE Loss across the scripts for each mask ratio R.

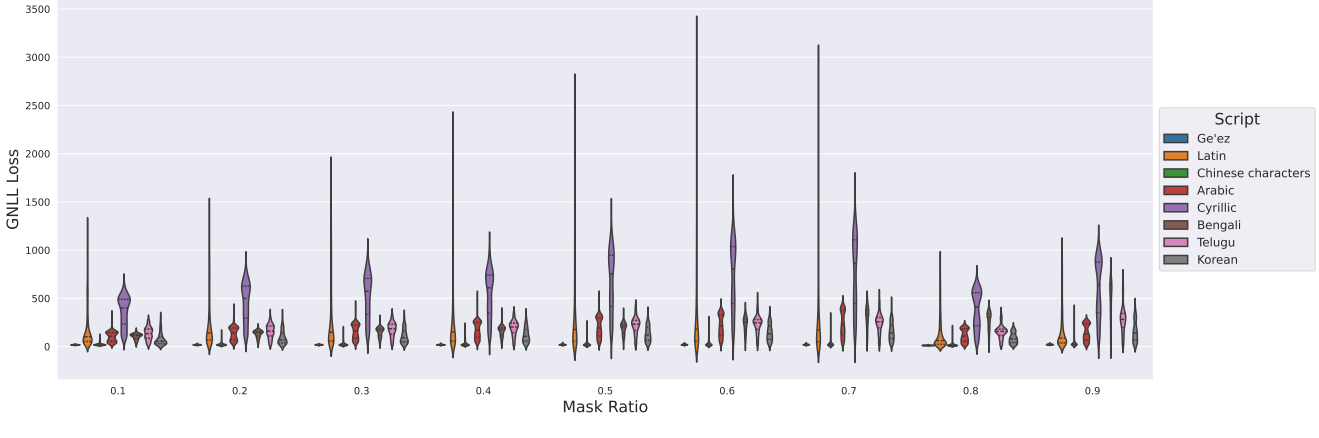


Figure 5.1.17: The distribution of the GNLL Loss across the scripts for each mask ratio R.

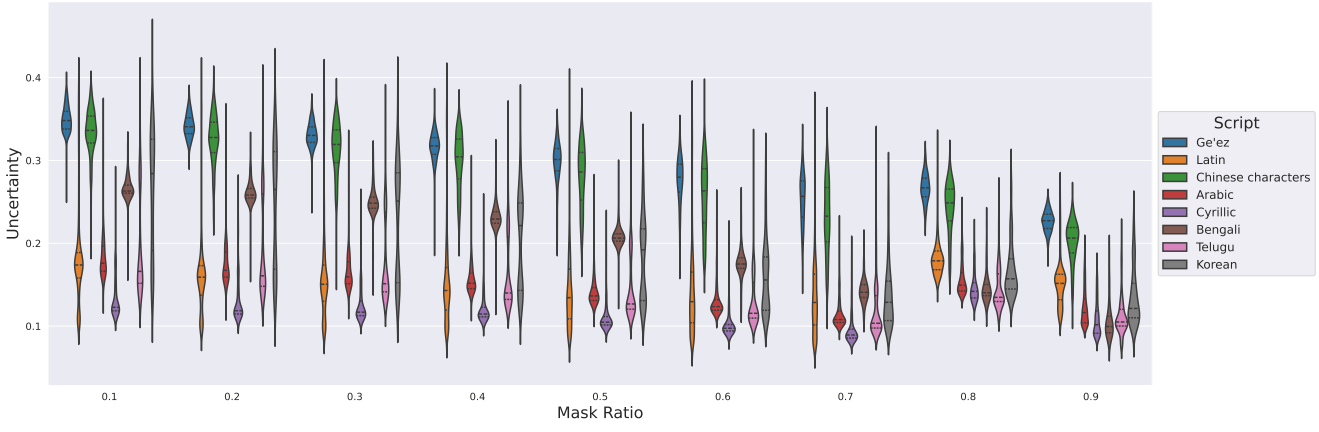


Figure 5.1.18: The distribution of the MC Uncertainty across the scripts for each mask ratio R.

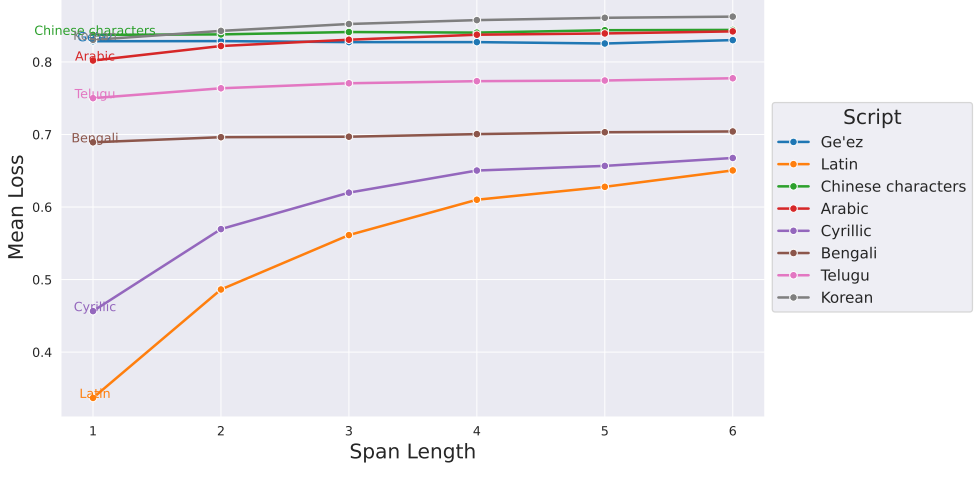


Figure 5.1.19: Mean MSE Loss across the different datasets for each span length S.

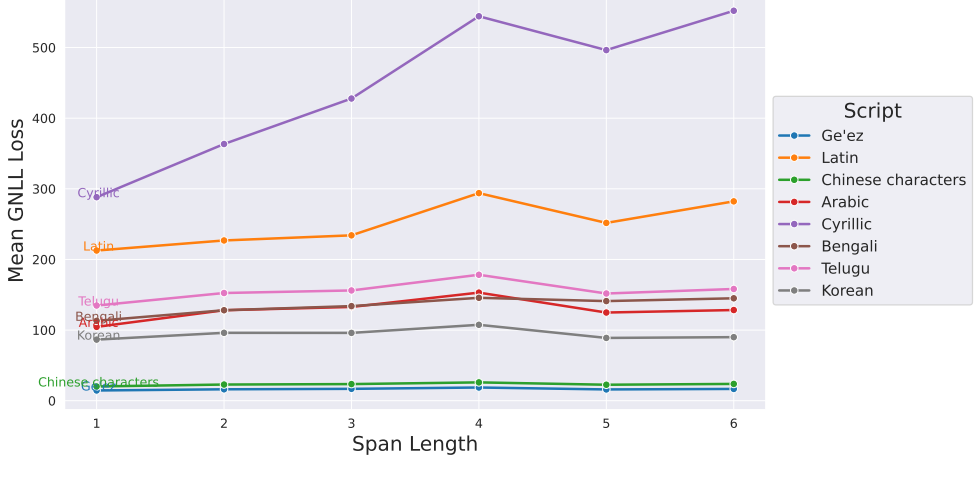


Figure 5.1.20: Mean GNLL Loss across the different datasets for each span length S.

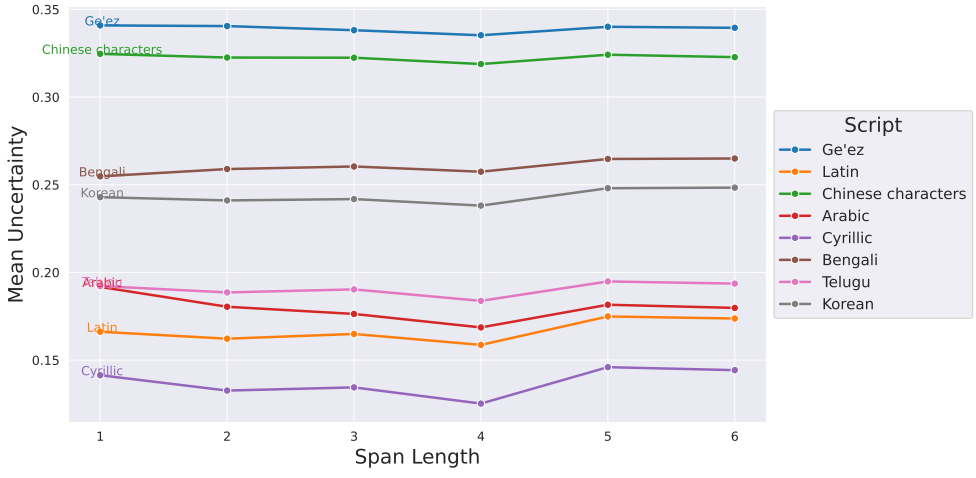


Figure 5.1.21: Mean MC Uncertainty across the different datasets for each span length S.

Figure 5.1.22: The distribution of the MSE Loss across the different datasets for each span length S.

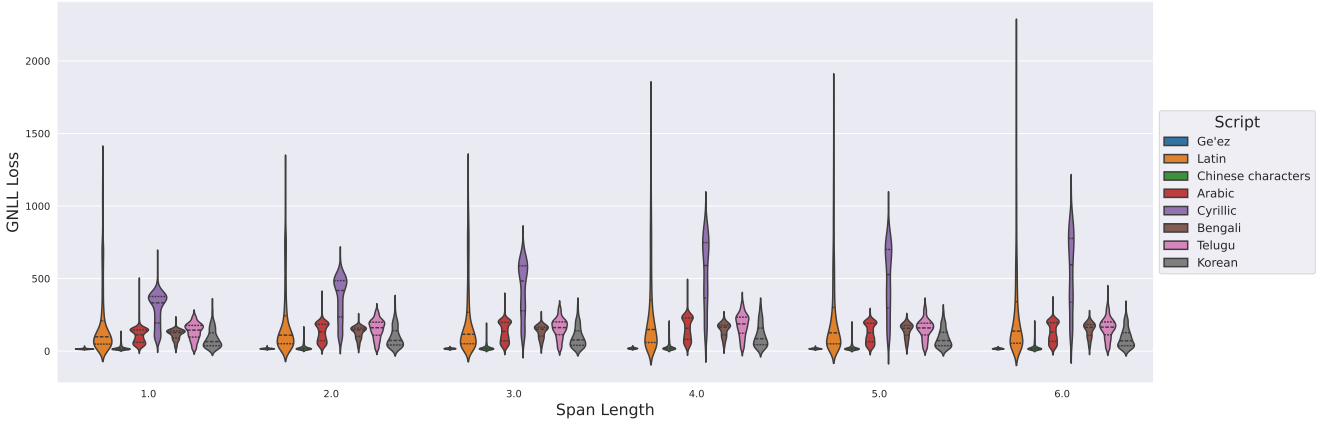


Figure 5.1.23: The distribution of the GNLL Loss across the different datasets for each span length S.

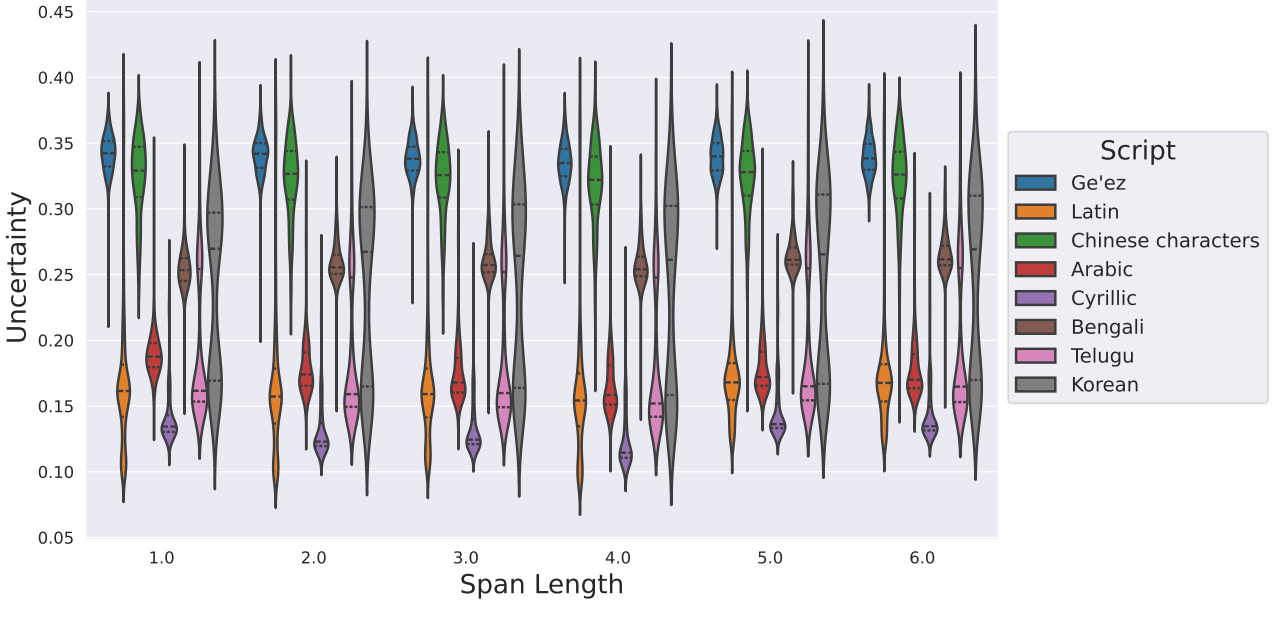


Figure 5.1.24: The distribution of the MC Uncertainty across the different datasets for each span length S.

5.1.3 Uncertainty Across Languages

Mask Ratio When studying the overall trends of loss and uncertainty across languages, there are clear differences between the groups. These can be seen in Figures 5.1.25, 5.1.26, and 5.1.27. The best-performing language is English (here the English dataset is an aggregation of multiple English examples from the different datasets), followed by Indonesian, and Finnish, which all use visually similar characters (Figure 5.1.25). The pixel-level uncertainty (Figure 5.1.27) is low (below 0.15) for these languages, demonstrating that the model can generalize to different languages within the same script. Not the same can be stated for languages such as Korean, Chinese, and Amharic, which show increased uncertainty, as well as reduced performance (with a MSE loss of over 0.8). The rest of the languages are situated in the middle of the range. Regarding the effect of mask ratio, the variation in predictions appears to decrease with the increase in mask ratio (Figure 5.1.27), suggesting that the model becomes more

confident that it does know the correct answer when large segments of the image are masked. An analysis of the results regarding the distribution of the different languages is not included, given that most relevant patterns can be identified by looking at the distribution of the larger groups.

Span Length Figures 5.1.28, 5.1.29, and 5.1.30 show the main findings for the span length experiment in terms of the different languages. Increasing the span length to $S \geq 5$ leads to an increase in the MSE loss (Figure 5.1.28) for one group of languages, while the rest of the languages stay constant. The second group contains languages which are more challenging for the model to reconstruct, such as Chinese Characters, Korean, and Bengali. Telugu and Bengali do appear to pose fewer difficulties than Korean, Arabic, or Chinese Characters, although they all use characters that are very different from Latin characters. The results on MC uncertainty in Figure 5.1.30 suggest that the level of uncertainty is not influenced by changes in span length, particularly for values of $S \leq 6$.

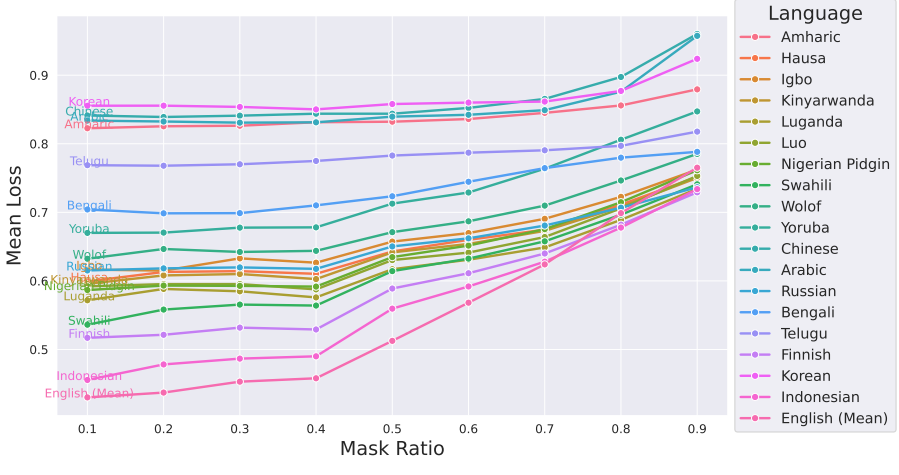


Figure 5.1.25: Mean MSE Loss across all examples for each mask ratio value R.

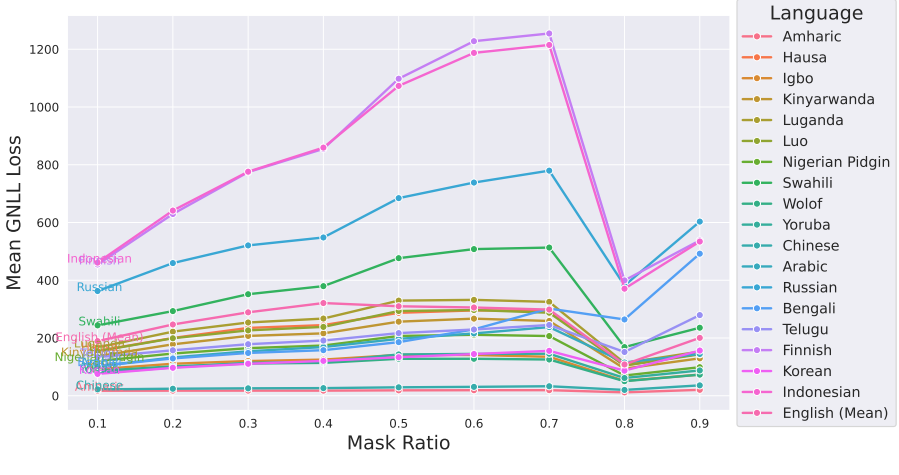


Figure 5.1.26: Mean GNLL Loss across all examples for each mask ratio value R.

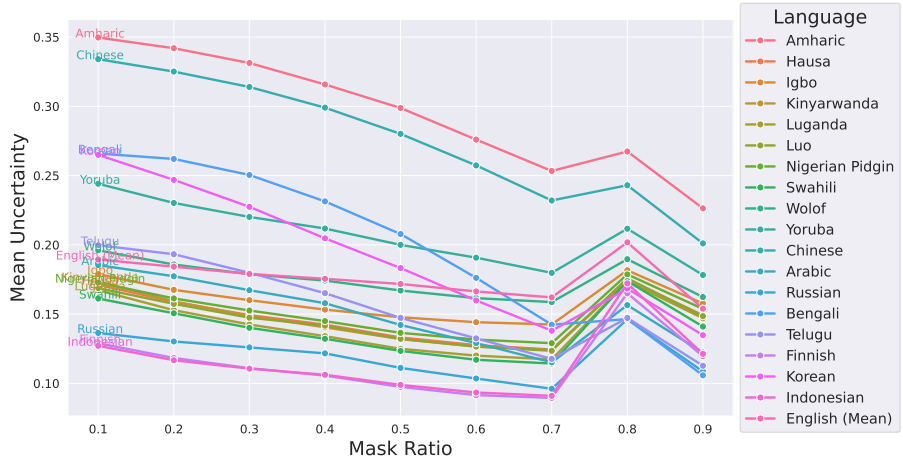


Figure 5.1.27: Mean MC Uncertainty across all examples for each mask ratio value R.

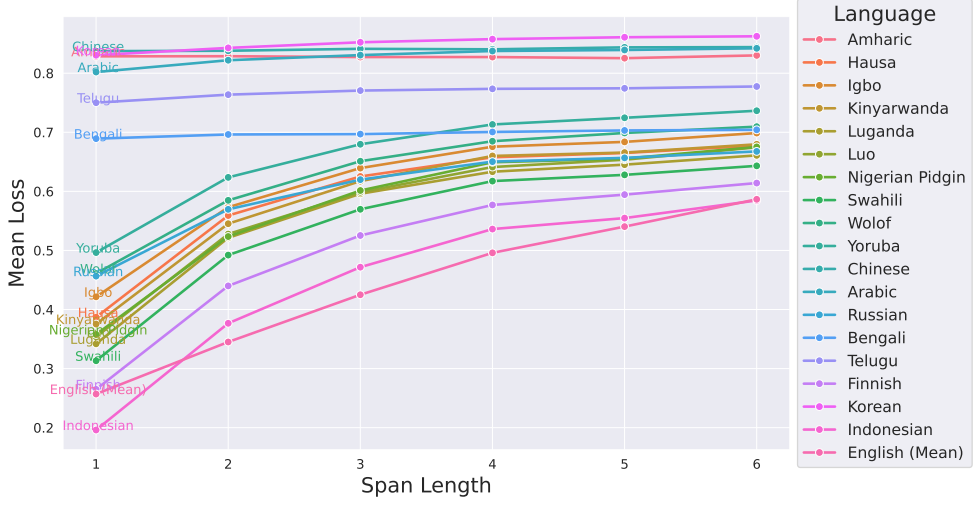


Figure 5.1.28: Mean MSE Loss across the different datasets for each span length S.

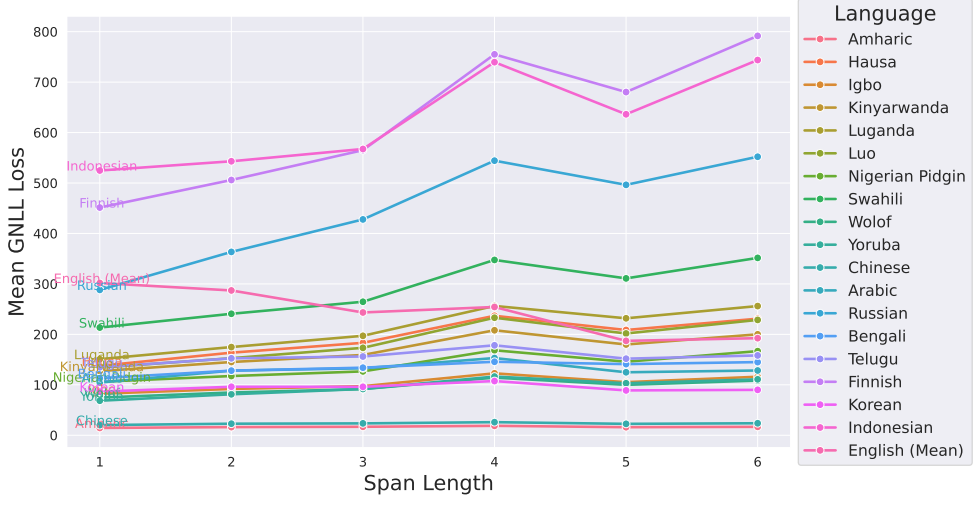


Figure 5.1.29: Mean GNLL Loss across the different datasets for each span length S.

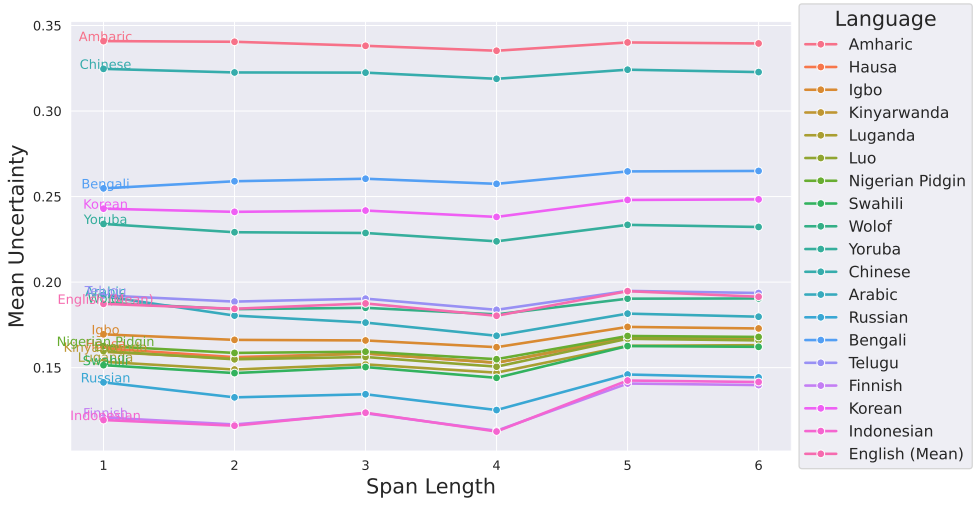


Figure 5.1.30: Mean MC Uncertainty across the different datasets for each span length S.

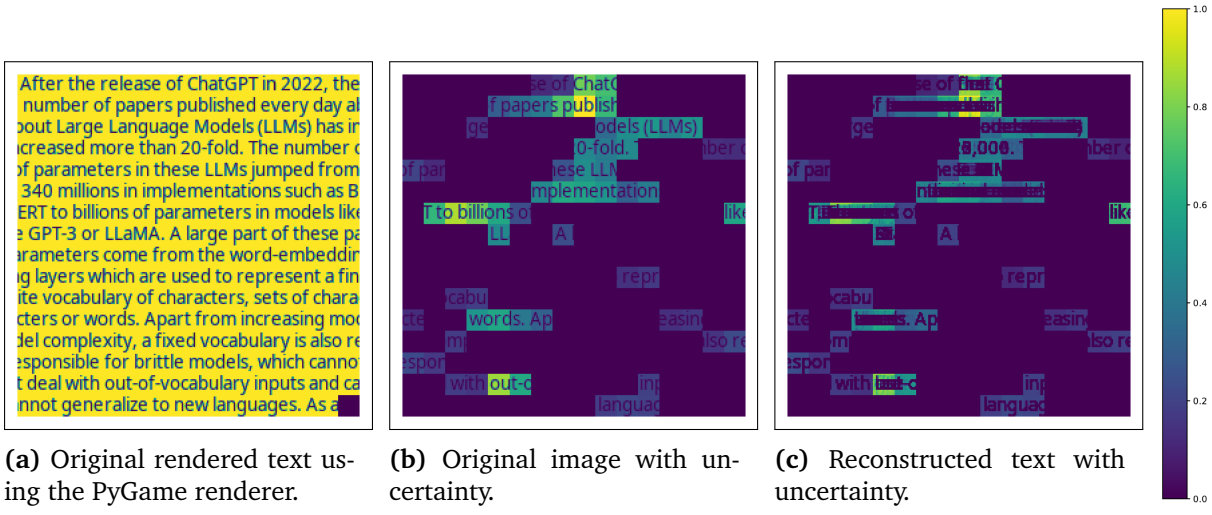
5.1.4 Visualizing Uncertainty in Text Reconstruction

To visualize MC uncertainty at the patch level, the current experiment looks at various text samples, including the best (Figure 5.1.33) and worst (Figure 5.1.32) performing samples in terms of the GNLL loss (Equation 3.4.3) across all datasets (see Table 4.1.1 for an overview of the experiments). Additionally, uncertainty is also visualized for a new piece of text, taken from the proposal of this thesis (Figure 1.2.2).

Figure 5.1.31 shows (a) the original rendered English text generated with the PyGame text renderer, (b) the original image overlaied with per-patch uncertainty and (c) the reconstructed text overlaied with per-patch uncertainty. Bright yellow patches suggest larger variations in predictions. This can be observed in the larger masked segments of patches from the first 6 lines of the image, as well as in lines 12 and 15. These segments also translate to less accurate reconstructions, as seen on the corresponding rows of the reconstructed image. On the other hand, smaller segments of patches (which appear darker in the image) are associated with lower uncertainty and are reconstructed more accurately. These patches often contain shorter sequences of letters. In terms of the mistakes, the model fails to reconstruct patches with numerals, such as *20-fold*. Still, it appears to understand that the most suitable prediction given the context is a number (the model predicts *20,000*). Moreover, longer and less frequent words such as *implementation* and *publish*, as well as punctuation marks (used in *(LLMs)*) appear to produce more variation in the prediction, given the increased uncertainty.

Figures 5.1.32 and 5.1.33 present per-patch uncertainty for the worst and best performing examples across all languages. This is calculated with respect to the GNLL loss, which penalizes the model based on the error and the prediction uncertainty. The most challenging examples come from Igbo, Swahili and Chinese, which is in line with the results found in the language analysis (see Section 5.1.3). All of these examples come from the MasakhaNER dataset and include mostly punctuation marks and numbers, suggesting that the model does not encode enough relevant features to correctly reconstruct these symbols. The top 5 performers list contains 4 examples from English and one from Nigerian Pidgin. This is in line with the findings from Figures 5.1.25 – 5.1.27. In the case of Nigerian Pidgin (first row in Figure 5.1.33), several unknown glyphs are

being rendered incorrectly, resulting in segments of text that might resemble English words, given that Nigerian Pidgin is an English-based creole language.



(a) Original rendered text using the PyGame renderer.

(b) Original image with uncertainty.

(c) Reconstructed text with uncertainty.

Figure 5.1.31: Example of uncertainty quantification at the patch-level for an image containing text from the proposal of this thesis. The uncertainty is measured as the standard deviation for each patch after Monte Carlo Dropout. Brighter colors indicate more uncertainty.

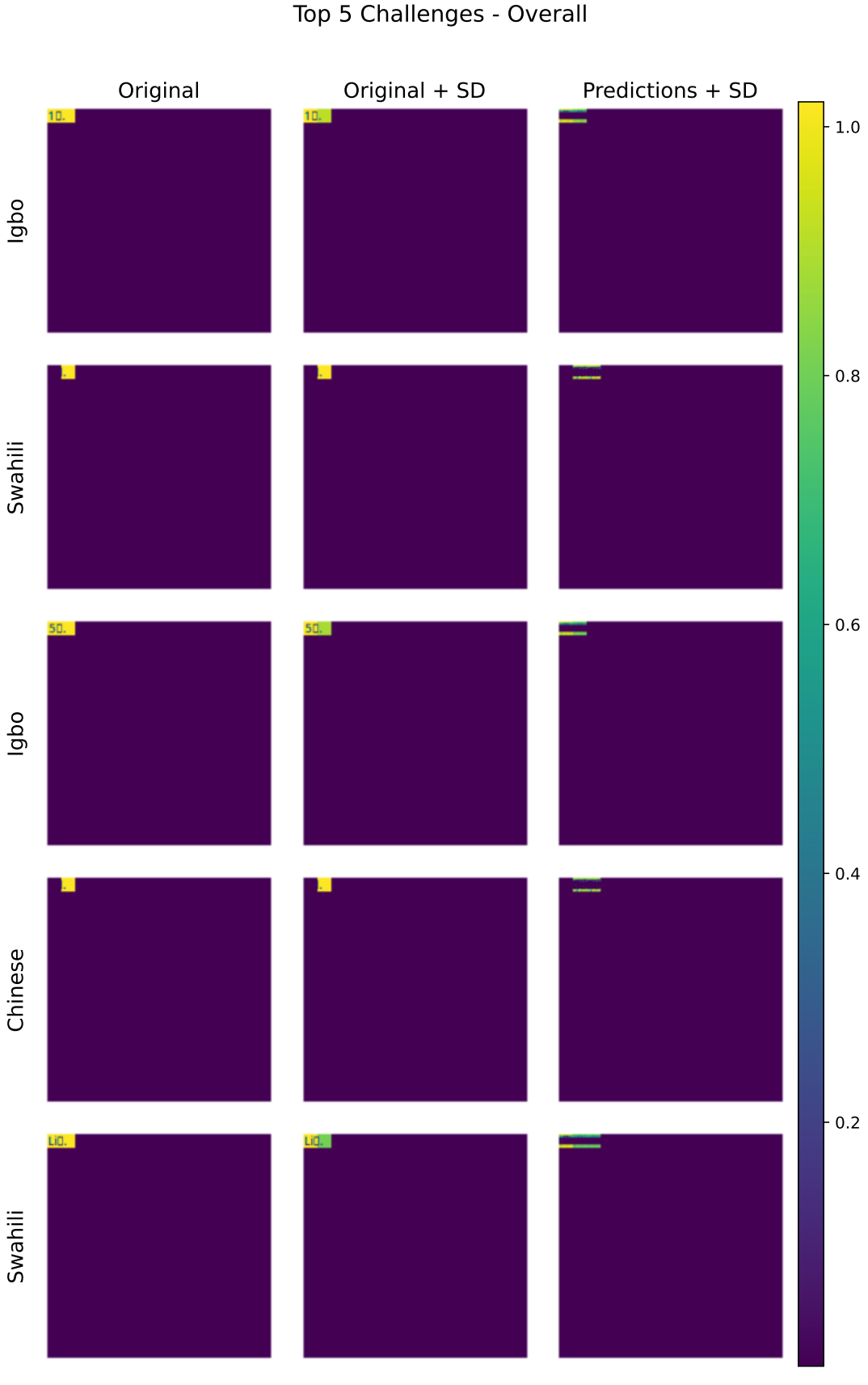


Figure 5.1.32: Top 5 challenges in terms of the GNLL loss across all datasets and languages. Only the encoded patches are shown. Brighter colors are an indicator of increased uncertainty. All pixels are normalized across all 15 images and scaled to the $[0,1]$ range.

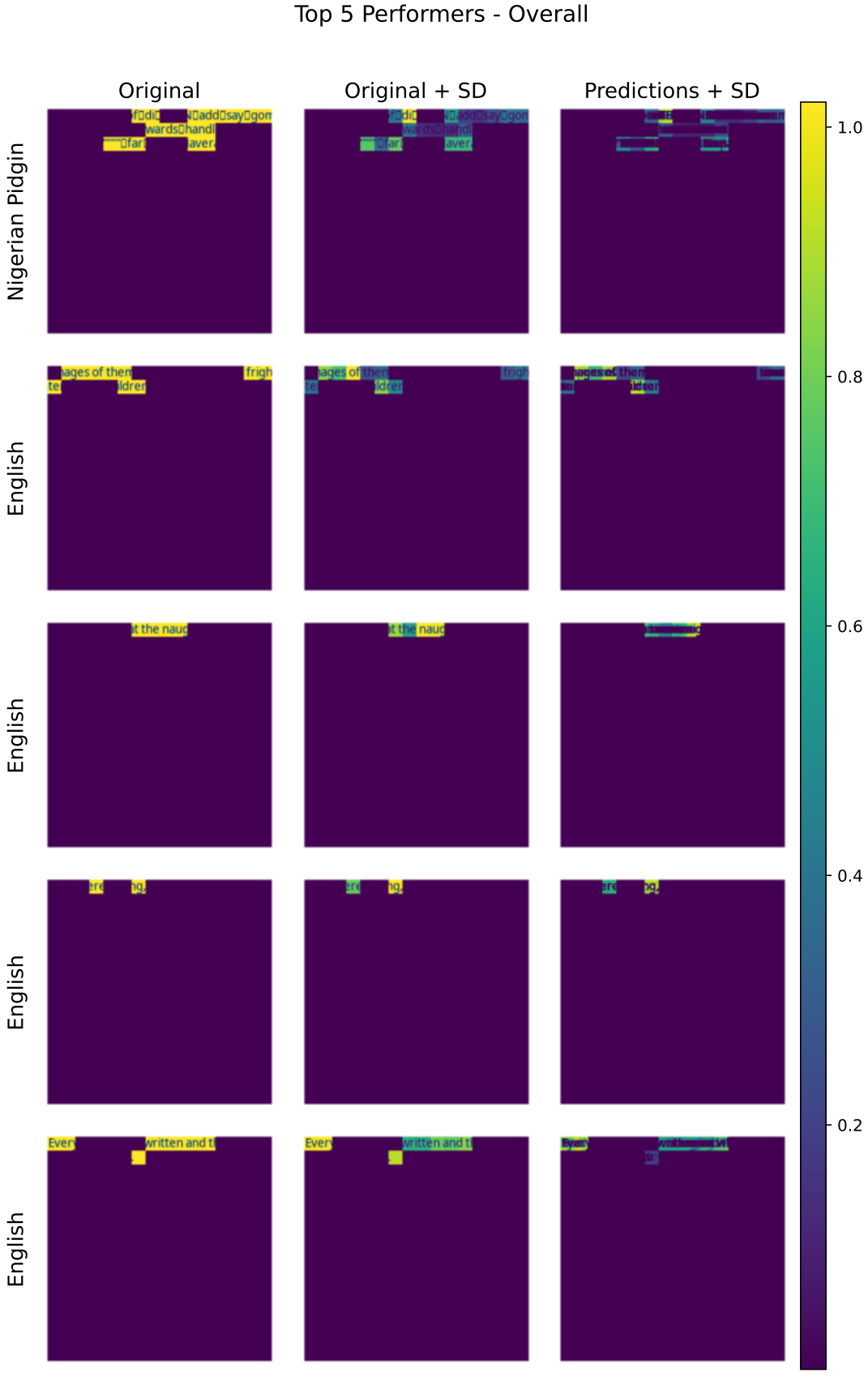


Figure 5.1.33: Top 5 performers in terms of the GNLL loss across all datasets and languages. Only the encoded patches are shown. Brighter colors are an indicator of increased uncertainty. All pixels are normalized across all 15 images and scaled to the $[0, 1]$ range.

5.1.5 Calibration Analysis

To further study the relationship between performance and uncertainty, Figure 5.1.34 depicts a hexbin plot with marginal distributions, where the Root Mean Squared Error (RMSE) loss is plotted against the SD uncertainty from the MC experiments. This variation of the calibration plot evaluates the calibration of uncertainty estimates in a regression task, which in this case is pixel value prediction. The x-axis represents the aggregated per-image standard deviation (uncertainty) of the model after 100 Monte Carlo samples. The RMSE measures the average of the actual errors between the true pixel values and the predicted values. Inside each hexagon, the color intensity corresponds to the density of data points within that hexagon. Therefore, darker regions indicate a higher density of data points. Figure 5.1.34 shows that there is a high density of points in the top left corner, which suggests that the model underestimates its performance. In other words, many examples are associated with high loss but low uncertainty.

The distribution of the points for all three datasets (MasakhaNER, TyDiQA-GoldP, and GLUE) is shown in the calibration plot from Figure 5.1.35. The highest level of overconfidence is associated with the question-answering task in TyDiQA-GoldP. However, there seems to be a subgroup of points for which the uncertainty is high. The points in the MaskhaNER dataset fall under the category of high uncertainty and high loss. The GLUE data is located between 0.15 and 0.3 on the uncertainty range and contains several examples showing decreased loss. While the model can be considered to be underestimating uncertainty with this group, the majority of the data still fall over the main diagonal, indicating an underestimation of uncertainty.

5.2 Attention Vizualization

The attention weights of the model are visualized across all 12 heads of the 12 layers of the encoder. The examples selected are Nigerian Pidgin as the best performing example and Igbo as the worst samples in terms of the GNLL loss according to Section 5.1.4. The results can be observed in Figure 5.2.1 and 5.2.2. Each cell in the attention grid shows the attention weights for the first 16 patches of a specific head h and layer l. In other words, given a selected patch in each column, the cell contains the attention from that patch to the rest of the patches in the sequence. The figures zoom in on two randomly selected cells ($L = \{2, 11\}), L = \{3, 5\}$) for a clearer comparison.

Overall, the results indicate that there are both differences and similarities in the kind of information that is encoded in the attention weights when comparing Igbo with Nigerian Pidgin. For both languages, the first four layers appear to encode the highest amount of visual information, given the high activation of the patches. Across all heads and layers of both examples, the attention weight corresponding to the CLS patch is high, as it contains the aggregate representation of the input patch sequence. There is a clear difference in the distribution of attention between the examples. The top 1 performer (Nigerian Pidgin) exhibits high activation on the diagonal at the neuron level, meaning that patches are attending to themselves, possibly to retain positional and contextual information. The Igbo example does not show the same pattern, rather a subset of dominant patches attend to the remaining ones.

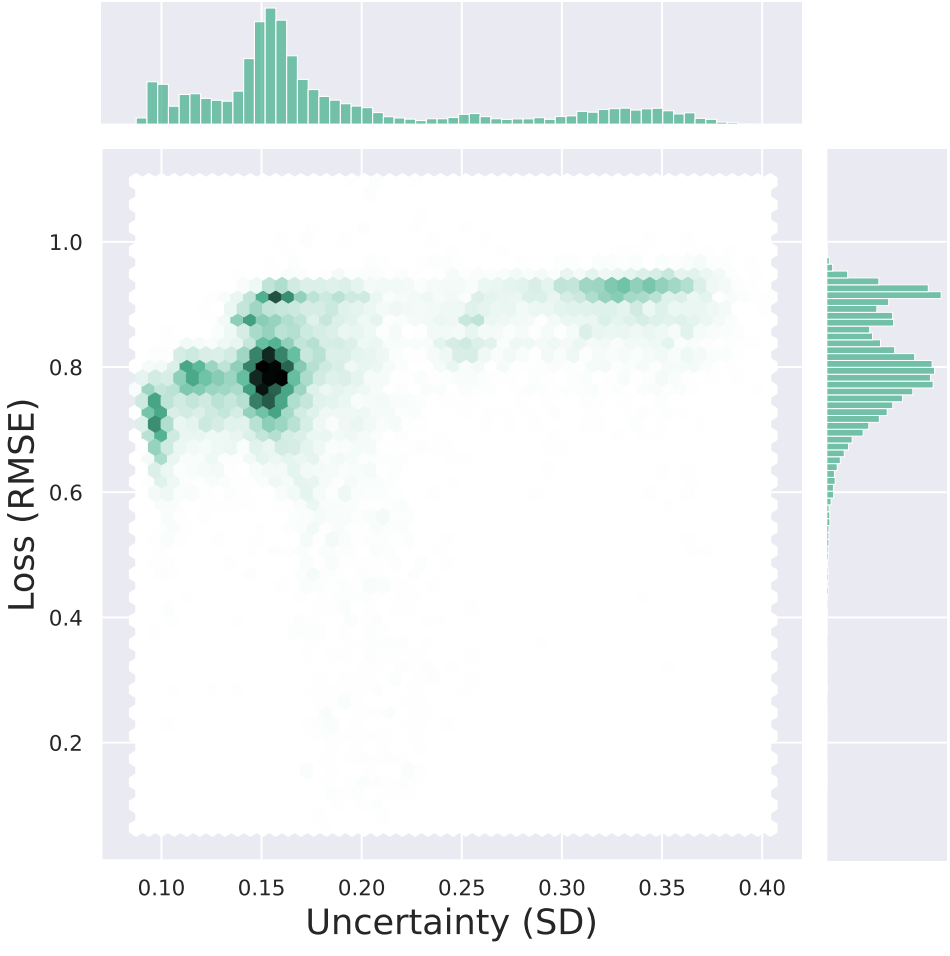


Figure 5.1.34: Calibration hexbin plot showing the RMSE loss in terms of the MC uncertainty.

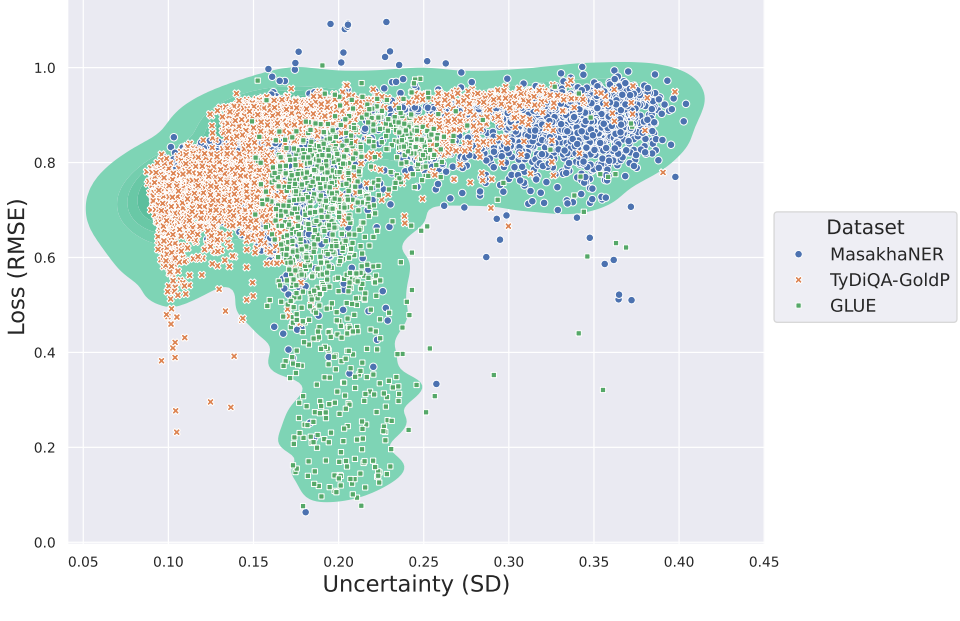


Figure 5.1.35: Calibration kernel density estimate plot showing the RMSE loss in terms of the MC uncertainty across the three datasets.

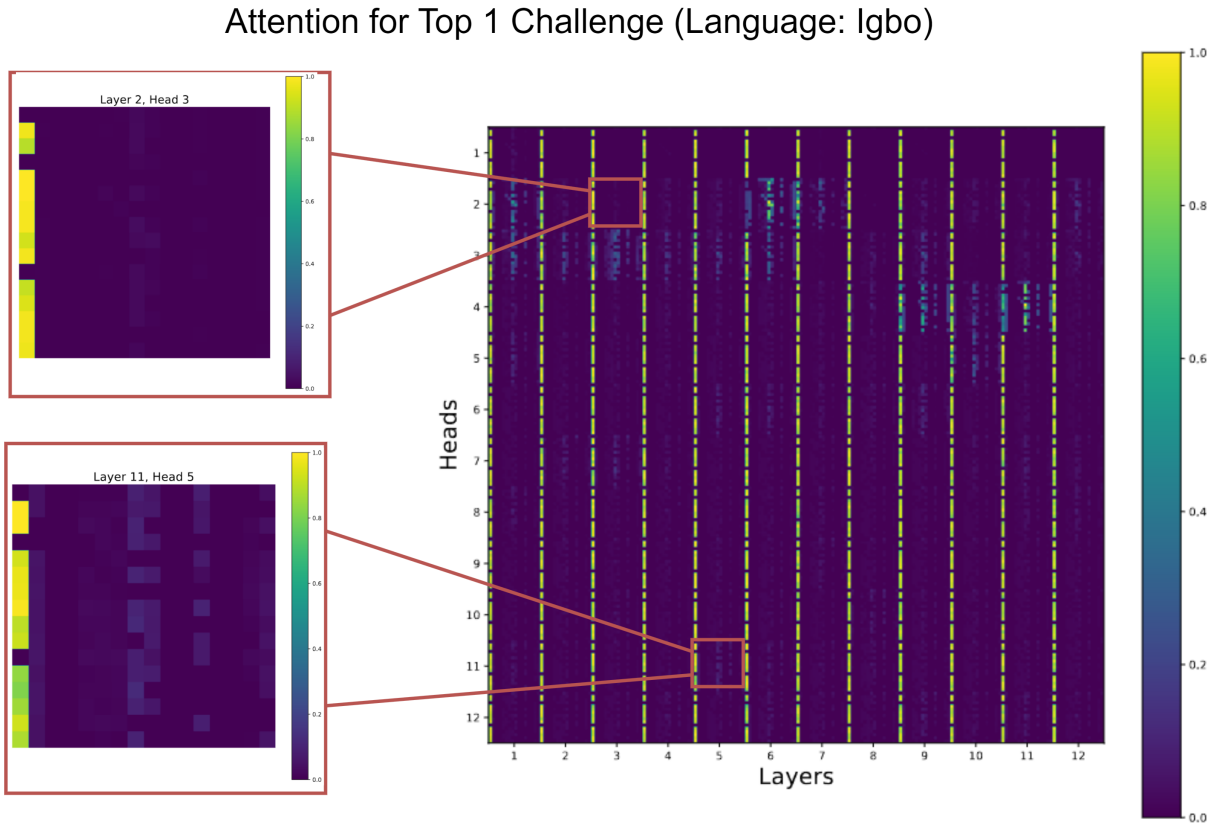


Figure 5.2.1: Model-level and neuron-level views of attention in the PIXEL model for the top 1 challenge in terms of the GNLL loss across all datasets.

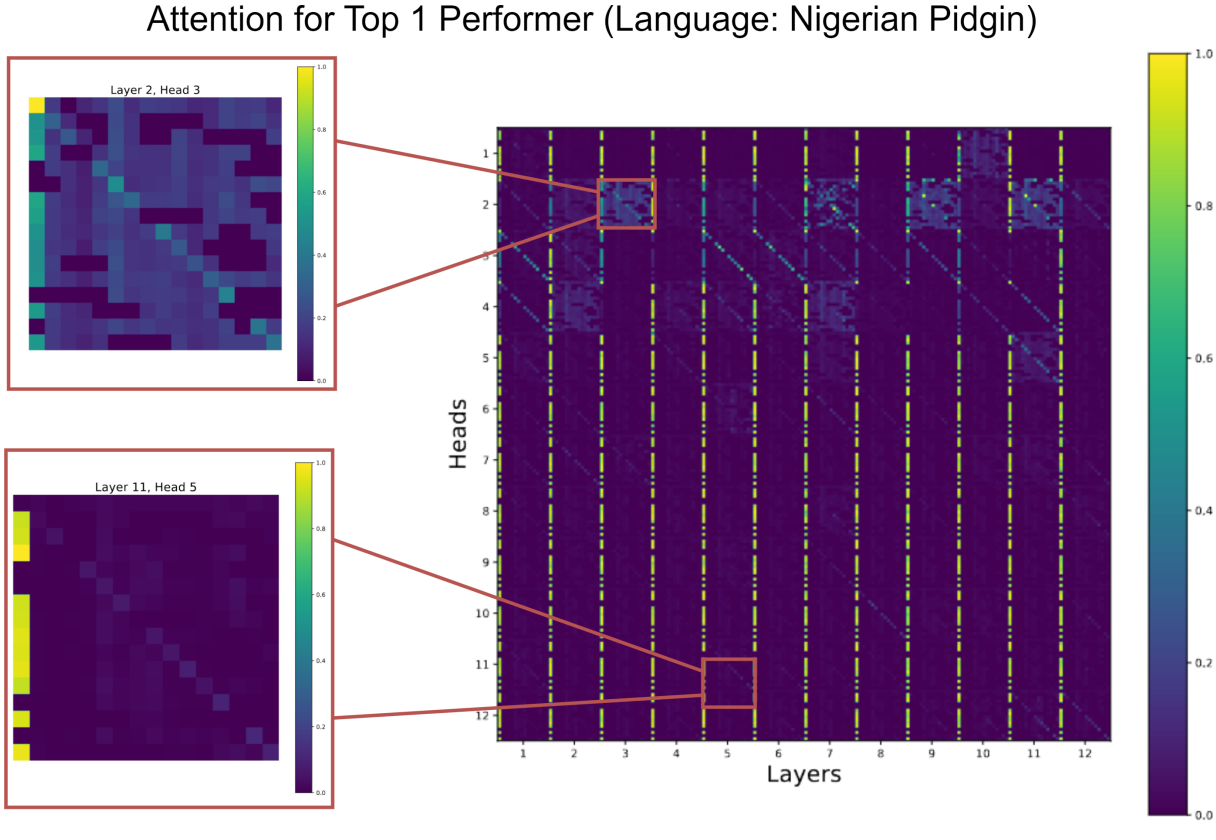


Figure 5.2.2: Model-level and neuron-level views of attention in the PIXEL model for the top 1 performer in terms of the GNLL loss across all datasets.

	ARA	BEN	FIN	IND	KOR	RUS	SWA	TEL	ENG	**AVG**
PIXEL	57.3	**36.3**	58.3	63.6	26.1	50.5	65.9	63.4	61.7	52.3
Ensemble	**59.5**	35.1	**59.6**	**67.3**	**27.1**	**53.3**	**67.1**	63.4	**62.1**	**54.0**

Table 5.3.1: The results of the QA task. The ensemble learning model finetuned on the TyDiQA-GoldP dataset is compared with the values reported by Rust et al. (2022). The metric shown is the F1 score, computed on the `validation` split of the data. The AVG score excludes ENG, as required (Clark et al., 2020).

5.3 Ensemble Learning

5.3.1 Extractive Question Answering

The results of the ensemble QA model are presented in Table 5.3.1, which shows the F1 score across all languages in the TyDiQA-GoldP dataset. These findings are compared with the results obtain by Rust et al. (2022), who finetuned the PIXEL model without ensemble learning. Overall, the ensemble learning method improves the performance in the extractive QA task for 6 out of the 8 languages. The average F1 score (excluding the ENG data) for the ensemble configuration is higher with 1.7 points than in the case of the regular PIXEL model. In terms of the individual languages, there is a high improvement for Indonesian (4.3 points), Russian (2.8 points), and Arabic (2.2 points), suggesting that combining multiple learners can improve performance regardless of script. At the same time, Telugu is associated with the same F1 score for both configurations, while the original PIXEL model performs better in Bengali.

Figure 5.3.1 presents the confidence distribution of the best answers in the ensemble model for all languages in the dataset. In general, the confidence is in the range $0.2 - -0.4$ across the majority of languages, with some distributions indicating slightly higher confidence, as in the case of Finnish, Indonesian, and Swahili. Lower confidence values can be seen in Korean and Bengali. These observations are in line with the previous findings on performance from Table 5.3.1. Figure 5.3.2 shows a clearer overview of the calibration of the ensemble QA model, measured in terms of the average F1 score and confidence. The red line indicates perfect calibration. All languages are placed above the line, suggesting that the finetuned model is slightly underconfident in the extractive QA task.

5.3.2 Named Entity Recognition

The results of the ensemble NER model are presented in Table 5.3.2, showing the weighted F1 score across the MasakhaNER 1.0 dataset. Due to hardware limitations at runtime, the *ENG* data is not included. For comparison, the results are shown against the values obtained by Rust et al. (2022). In general, ensemble learning improves the performance significantly for all 9 languages, resulting in scores > 90. This is also the case for languages that were previously associated with a low score, such as Amharic (*AMH*). The F1 score gap is 24.3 points in favour of the ensemble method, suggesting that ensemble learning improves the comprehension of long-term dependencies in NER tasks.

The series of plots in Figure 5.3.3 depict the confidence for each label and language in the NER classification task. The legends show the label counts for each class and it

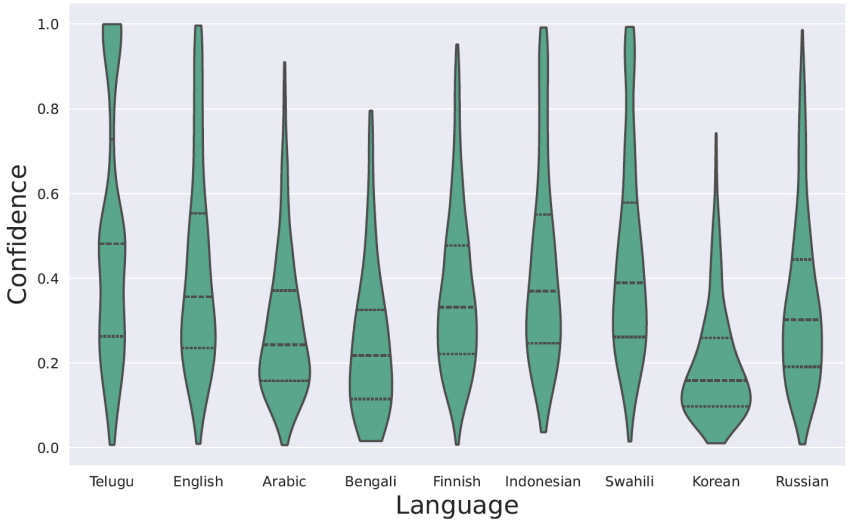


Figure 5.3.1: Confidence distribution across all languages in the TyDiQA-GoldP dataset for the ensemble model.

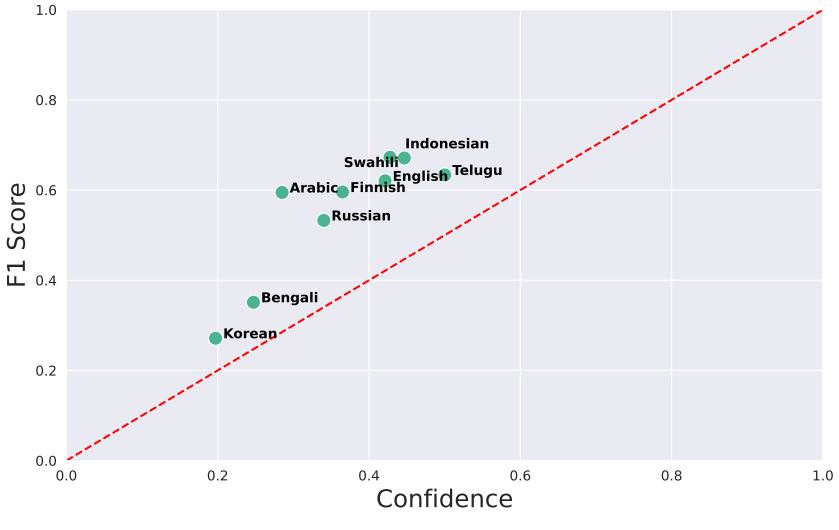


Figure 5.3.2: Calibration plot showing performance in terms of the F1 score versus confidence for the ensemble model finetuned on the TyDiQA-GoldP dataset.

is clear that the majority of words do not represent a named entity. Nevertheless, the confidence is very high, with mean values of over 0.8 for all languages. Across labels, the confidence associated with the $B-ORG$ and $I-ORG$ labels is lower in languages such as Kinyarwanda, Nigerian Pidgin, Swahili, and Wolof. Also, the model is less confident in identifying the inside of a location ($I-LOC$) than the beginning of a location ($B-LOC$). On the other hand, the $DATE$ entity appears to have a high confidence across all languages.

5.4 Results Discussion

The results in the MC Uncertainty experiment generally indicate high uncertainty for a high mask ratio. Still, the most optimal value is a mask ratio of 50%, representing a reasonable trade-off between uncertainty and loss. Overall, the SC task is linked to the highest level of uncertainty. The reason might be the amount of text rendered in the

	AMH	HAU	IBO	KIN	LUG	LUO	PCM	SWA	WOL	YOR	**AVG**
PIXEL	47.7	82.4	79.9	64.2	76.5	66.6	78.7	79.8	59.7	70.7	70.7
Ensemble	**90.2**	**97.1**	**96.1**	**93.9**	**95.5**	**93.1**	**97.1**	**96.1**	**95.8**	**95.2**	**95**

Table 5.3.2: The results of the NER task. The ensemble learning model finetuned on the MasakhaNER 1.0 dataset is compared with the values reported by Rust et al. (2022). The metric shown is the F1 score, computed on the `test` split of the data.

image, as the text in the GLUE dataset takes up more patches than single tokens in the NER task or single questions in the QA task.

Scripts such as Latin are less uncertain, indicating that multilingual pretraining is necessary but instead of language, one can focus on introducing a new script, as there is evidence to suggest that there exists knowledge transfer between scripts like Latin and Cyrillic. For example, finetuning on one language such as Chinese might benefit performance in other languages like Korean or Amharic. This approach is more robust than traditional LLMs, where the transfer of learning happens under stricter conditions, for instance when languages share syntactic structures or when there is a significant overlap between vocabularies.

In terms of the span length, there is not a big difference between shorter or longer sequences of patches. This might suggest that the masked autoencoder does not heavily rely on the extended context of the characters, but rather on the surrounding pixels to reconstruct a character. One explanation for this is the presence of morphemes within a language, which carry meaning at the word level. However, when visualizing the uncertainty for each patch, rather than as an average on the entire image, there are clear differences between short and long spans, with longer sequences being associated with higher uncertainty. Thus, different measures of quantifying uncertainty can yield different results.

Regarding calibration, the pretrained model appears to underestimate uncertainty at the patch-level. On the other hand, when finetuning ensemble models and evaluating them on downstream tasks like Named-Entity Recognition and Question-Answerting, the confidence in the predictions is increased and it matches the F1 score more closely.

Ensemble learning with parameter tuning shows very high performance during the selected tasks, compared to the single model. Moreover, this is achieved using a low number of individual learners, 4 in the QA task and 5 in NER. The ensemble is also robust when it comes to class imbalances in the NER task, where only 12% of the tokens are named entities. Some limitations of this method include the hardware and training time required to train multiple models. Nevertheless, PIXEL has 20% fewer parameters than BERT, so an ensemble of PIXEL models remains less complex than the BERT variant and significantly more lightweight than models like GPT.

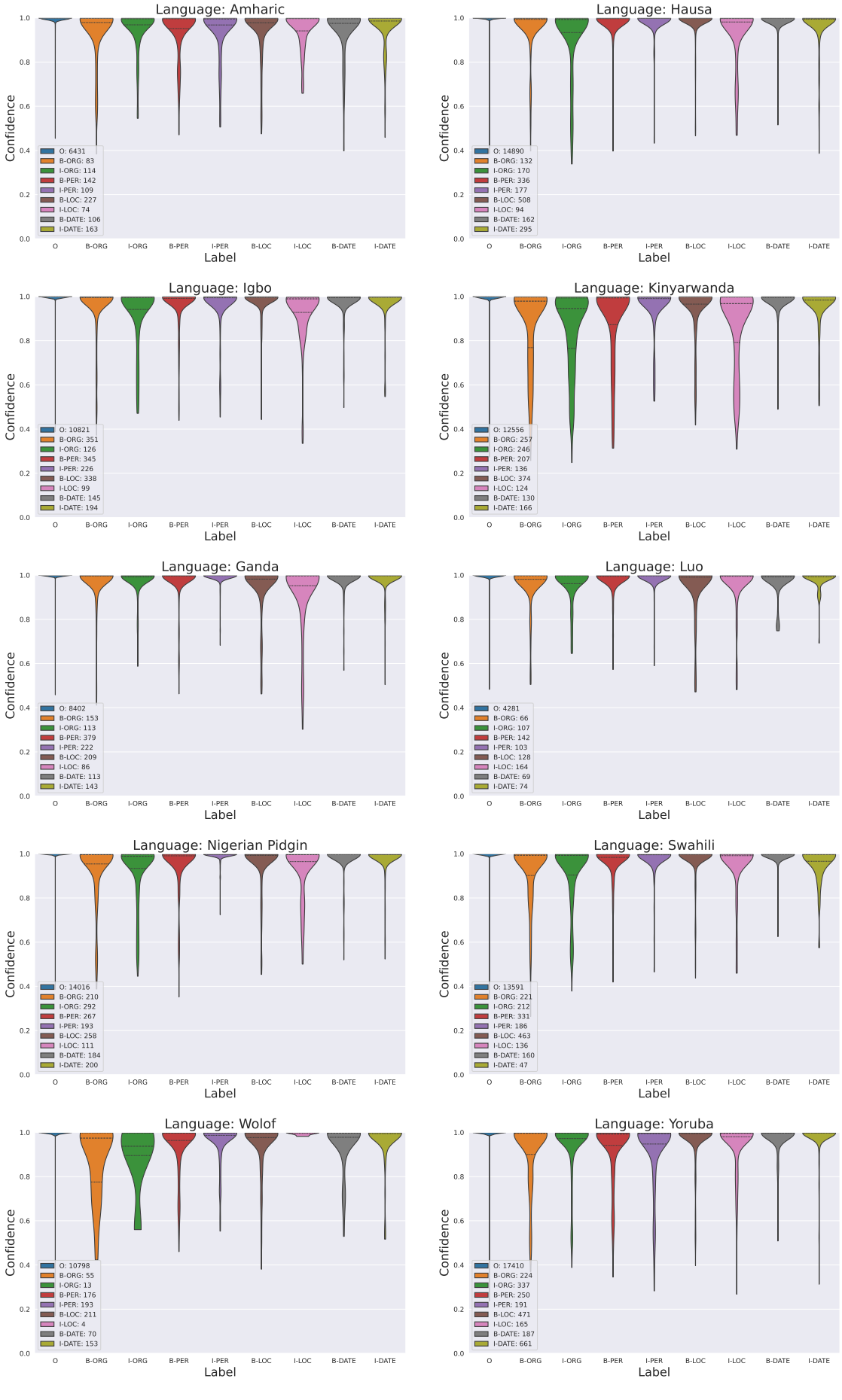


Figure 5.3.3: Confidence distribution of the ensemble NER model across languages.

6 Discussion & Conclusion

6.1 Discussion

In the beginning, this study asked 5 research questions. According to the findings and experiments, the answers are as follows:

1. Do visual language models represent a viable solution for the semantic processing of text? Yes, with some observations. The results of Rust et al. (2022) suggested that the PIXEL model finds semantic tasks more challenging than syntactic tasks and that further pretraining and additional inductive biases are necessary to close the performance gap between PIXEL and BERT. This work shows that further pertaining can be ditched in favor of ensemble learning to obtain better performance than BERT, even for languages that are visually dissimilar from the pertaining language. This comes with the additional benefit of improved model calibration, as well as reduced training time. Finetuning the same model is more efficient and robust compared to altering the base model. However, due to limited computation resources, the comparison on the GLUE dataset is missing, but given the performance in NER and QA, it is safe to assume that the ensemble framework will outperform the single-learner variant.

2. How can uncertainty quantification and calibration methods be integrated into visual language models? This work showed that it is possible to integrate uncertainty quantification methods and measure calibration in the context of visual text models. These methods include Monte Carlo Dropout at the patch level, with the observation that more work should be directed towards finding more effective ways of aggregating and visualizing uncertainty across longer patch sequences. Attention based methods can also be used to gain insights into how these models encode information, but there remains the debate about whether or not attention counts as an explanation (Bibal et al., 2022). Still, this debate falls outside the scope of this research. Ensemble learning with a low number of individual learners can also be used successfully within visual text models, to improve both performance and confidence.

3. How do visual language models encode uncertainty at the patch-level? This work employed Monte Carlo Dropout methods to study how uncertainty is encoded at the patch level. According to the results, uncertainty is dependent on the mask ratio value, with more masking leading to a more uncertain text reconstruction. The findings on the span length are mixed, partly due to limitations in the current approach, which are discussed in Section 6.2. The script of the pretraining language has a large influence on patch-level uncertainty. Latin and Latin-like languages are associated with lower uncertainty values, while scripts such as Korean or Arabic are more uncertain during reconstruction.

4. How is the attention mechanism represented in visual language models? While it is possible to visualize the attention mechanism in visual language models, there are some comments to be made about this. Unlike traditional language models like BERT where each token represents a meaningful unit and the relationship between two tokens can be understood intuitively, the patches in visual language models cannot be mapped back to text chunks. This makes it more challenging to interpret how attention is paid to

the different patches and what are the implications of these connections in the context of the entire model. Moreover, given the large number of attention structures and the image dimensions, visualizing attention for all patches simultaneously becomes very difficult.

5. What is the effect of ensemble learning in finetuning visual language models? Ensemble learning can be applied successfully to improve performance and calibration in visual language models. The evaluation shows higher F1 scores for 17 of the 19 tested languages across two tasks. The models become more robust and can overcome individual weaknesses by aggregating predictions from multiple learners using hyperparameter tuning. Additionally, ensemble learning improves calibration through better error diversification and data representation.

6.2 Limitations & Future Work

Limitations The current study is subject to several limitations. Firstly, the way uncertainty is computed at the image level during the MC experiments can be more reliable. At the moment, uncertainty is averaged across all pixels in an image. However, this does not account for the difference in span length, as some sequences of patches are longer than others. Quantifying uncertainty as an average for each span length in the image could bring more insights into how the model encodes long-term dependencies. Secondly, the information in the attention plots should be aggregated so that all patches are visible at once, while keeping a reasonable image size. Using the current method, visualizing all 256 patches across the 144 attention structures would result in a very large and difficult to interpret image. Regarding the calibration analysis, it is not completely clear that the two measurements of performance (loss vs. MC uncertainty during the pretraining stage and F1 score vs. confidence during finetuning) are quantifying the same underlying metric. For this reason, additional testing should be performed to establish the exact effect size of ensemble learning on model calibration.

Future Work One point to be explored in future works on text reconstruction is the idea of pixels-as-tokens in the context of the Pixel Transformer (PiT) model, introduced by Nguyen et al. (2024). Instead of training the model to perform patch reconstruction, PiT treats each pixel as a token and the reconstruction happens at the pixel level. Evidence suggests that this method completely removes locality as in inductive bias. This can potentially improve long-term context comprehension in the proposed approach, as the current findings indicate that the reconstruction of characters depends on neighboring pixels. Additionally, the finetuning pipeline can be expanded to more complex semantic tasks, such as summarization, open-ended question answering where the answer is not always explicitly mentioned in the context, and text generation (Li et al. (2023) introduced a new method for text generation using GlyphDiffusion). To improve model calibration, post-hoc methods like temperature scaling can be used either separately or in combination with Monte Carlo (Laves et al., 2019). During pretraining, the Cross-Entropy loss can be replaced by the Focal Loss, which is effective in calibration models trained on imbalanced datasets (C. Wang et al., 2022).

6.3 Summary

Motivation This thesis focused on studying a solution found at the intersection of three problems in traditional Language Modeling. The semantics problem (Section 2.4) makes it challenging for models to maintain context over long text spans. The vocabulary problem (Section 2.5) leads to a bottleneck when new words have to be added to the pool. The uncertainty problem (Section 2.6) is caused by unreliable models, which are overconfident and lack proper calibration. In this context, Visual Language Models that use rendered text as input (not to be confused with models that learn from both text and image data) can be applied to tackle these limitations. The main aim of this work is analyzing various uncertainty and confidence quantification methods in the PIXEL model, motivated by the very reduced number of studies on the topic.

Methods & Results These methods included Monte Carlo Dropout (Section 3.4.1) to measure uncertainty at the patch level in the pretrained model. A series of experiments looked at uncertainty distribution across the mask ratio and span length hyperparameters, as well as across 3 semantic tasks (Named Entity Recognition, Sequence Classification, and Question Answering), 7 scripts, and 18 languages. According to the findings of Section 5.1, a large mask ratio leads to more uncertain reconstructions, while the effect of the span length is not yet clear. Scripts that are visually different from Latin, as well as semantically challenging tasks such as Question Answering are also associated with higher uncertainty. Visualizing the Attention Mechanism (Section 3.4.2) can also provide insights into how the ViT encodes patch information. To achieve a model and neuron view of attention, the attention weights are shown for the first 16 patches of an image across the 12 attention layers and 12 heads. Ensemble learning (Section 3.4.3) was used to train multiple learners with different parameter configurations on the MasakhaNER 1.0 and TyDiQA datasets. The results (Section 5.3) indicate that this method improves F1 performance across most tested languages. In terms of calibration, the pretrained model appeared to be overconfident when reconstructing text. The calibration improved during finetuning with ensemble learning.

Conclusion The findings of this study indicate that Visual Language Models represent a viable and lightweight solution to traditional language modeling, even for tasks that require semantic understanding of text. The reliability and explainability of VLMs can also be improved through uncertainty quantification methods, as shown during the experiments. Future research should focus on perfecting the existing techniques and exploring new ways of understanding the inner workings of VLMs.

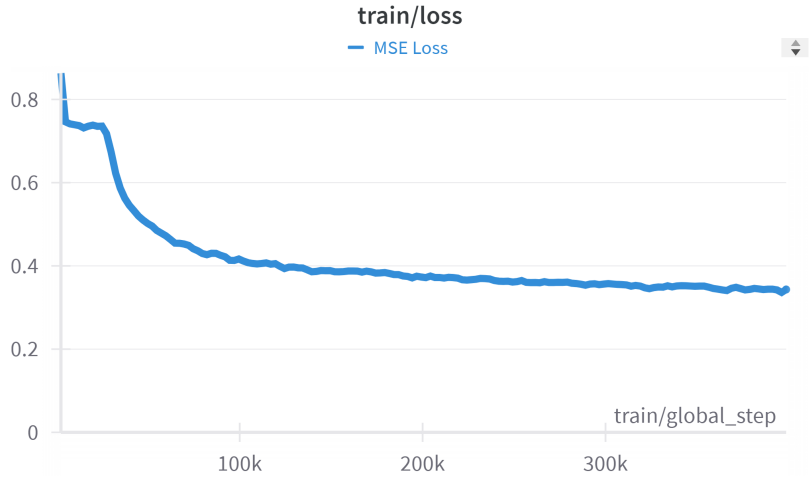


Figure 7.1.1: The training loss (MSE) of the multilingual PIXEL model.

7 Appendix

7.1 Multilingual Pretraining

This study also proposes a multilingual pretrained model based on the PIXEL architecture. This uses the PangoCairo backend, which supports both left-to-right and right-to-left scripts. To enhance context comprehension, the model uses bigram patch embeddings instead of unigram patch embeddings, meaning that the embeddings are created based on pairs of consecutive patches, and not based on single patches. To preserve the vector size of 768, the embeddings for the two patches inside a pair are averaged to create one vector. Other aggregation methods are also available in the implementation, such as vector concatenation, or applying a linear projection.

Due to a lack of computation resources, the model was trained for 3 days only (The English version needed 8 days according to Rust et al. (2022)), on a dataset consisting of 18 rendered languages, as shown in Table 3.1.1. The rendered datasets are available at `https://huggingface.co/stefania-radu`. The training loss can be seen in Figure 7.1.1, showing that this method carries potential. However, more training and rigorous evaluation are needed to establish how the multilingual approach compares to the single-language version.

7.2 Examples of Reconstruction with Uncertainty

Additional examples of visualized text reconstruction with uncertainty can be seen in Figures 7.2.1 and 7.2.2. The text is randomly selected from the Wikipedia dataset of each language (`https://huggingface.co/datasets/wikimedia/wikipedia`). The first column shows the original rendered image, the second column contains the non-masked patches with uncertainty (Monte Carlo standard deviation), and the final column is the reconstructed text (mean prediction) with uncertainty. Bright patches indicate higher uncertainty.

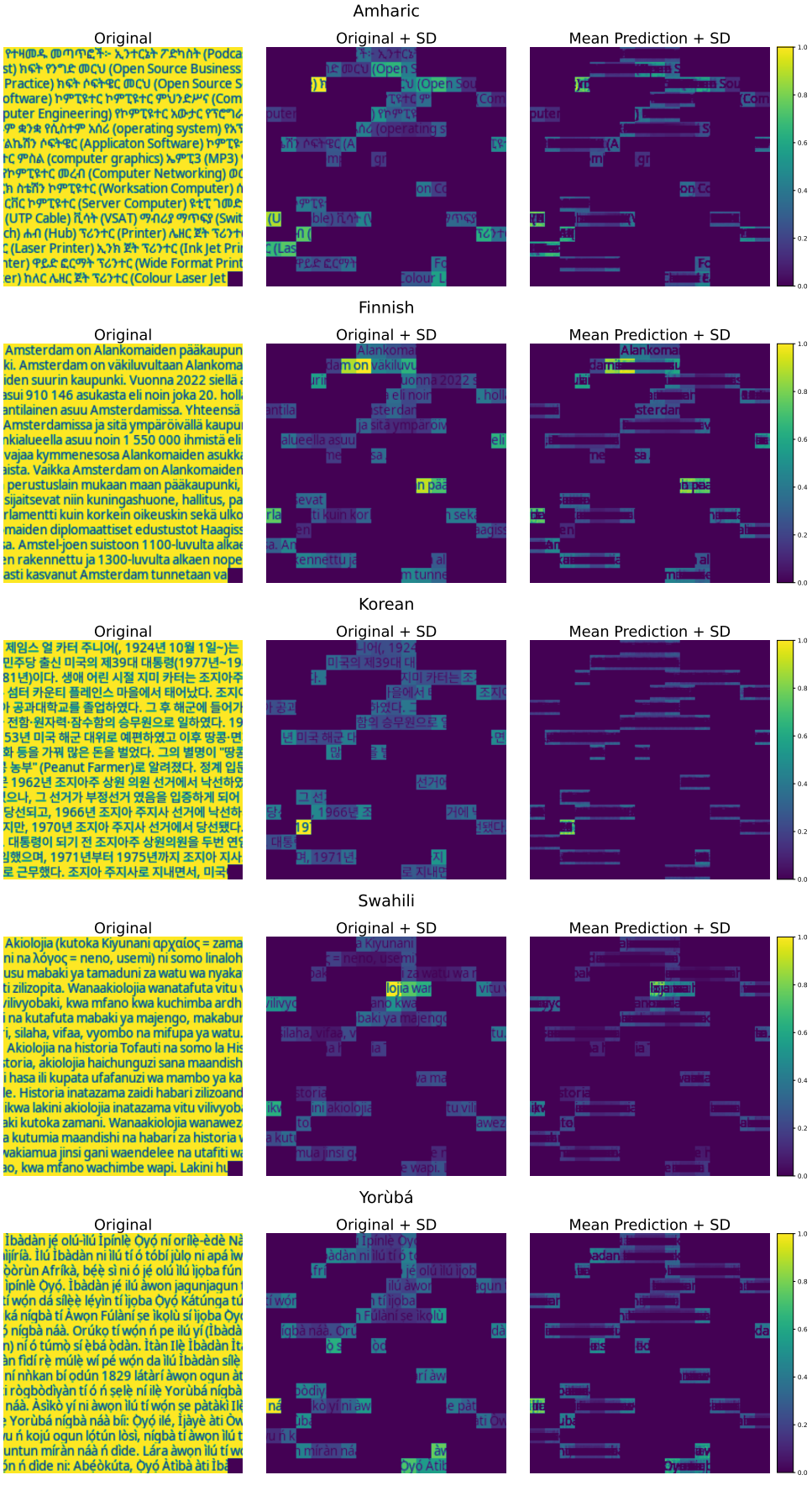


Figure 7.2.1: Examples of uncertainty quantification at the patch-level for various languages.

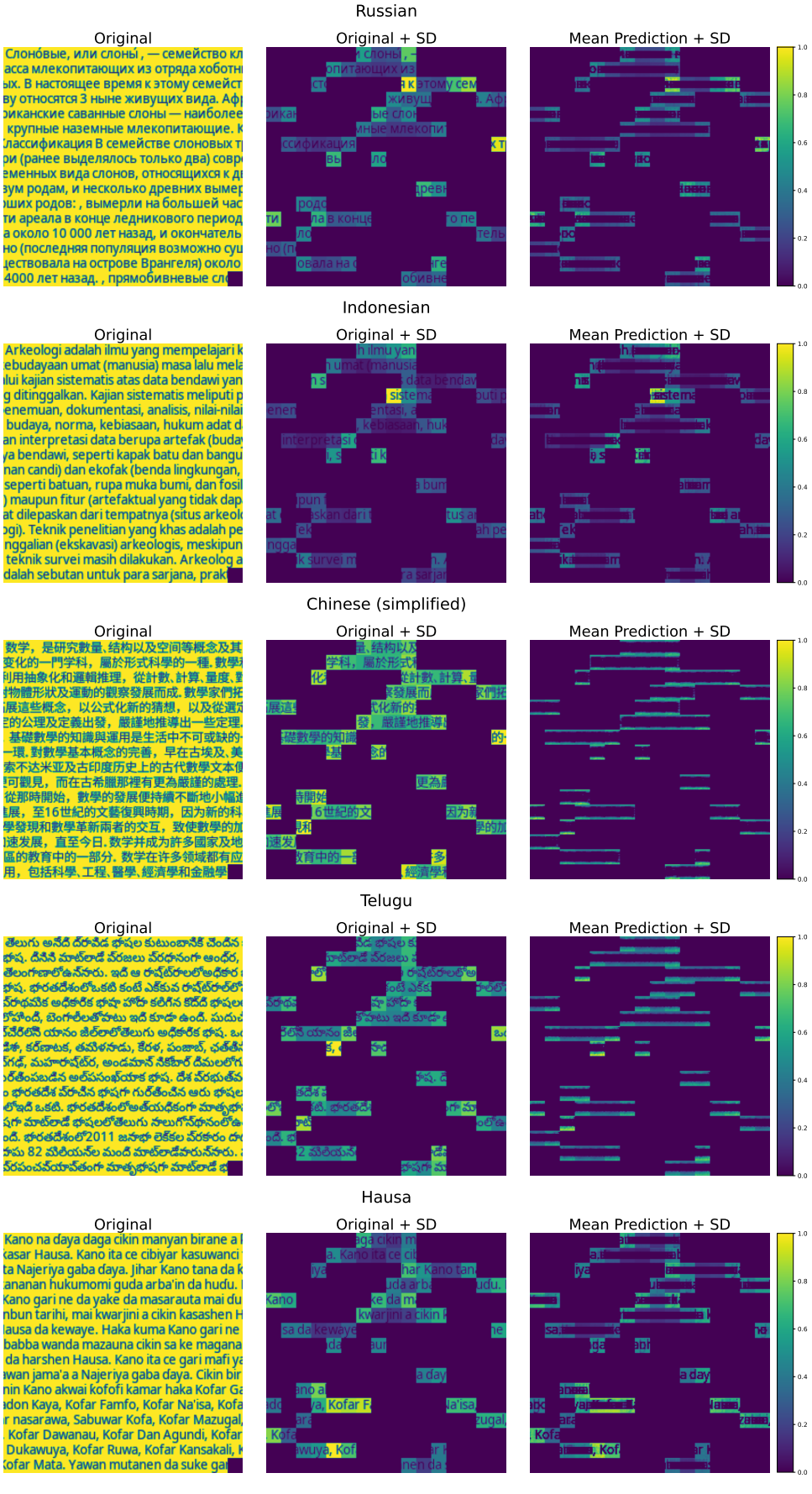


Figure 7.2.2: Examples of uncertainty quantification at the patch-level for various languages.

7.3 Code

The complete implementation can be found at `https://github.com/stefania-radu/pixel-semantic`, which extends the PIXEL repository of Rust et al. (2022). The code is based on PyTorch (Paszke et al., 2019) and HuggingFace transformers (Wolf et al., 2020).

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... others (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In *12th usenix symposium on operating systems design and implementation (osdi 16)* (pp. 265–283).

Adelani, D. I., Abbott, J., Neubig, G., D'souza, D., Kreutzer, J., Lignos, C., ... others (2021). Masakhaner: Named entity recognition for african languages. *Transactions of the Association for Computational Linguistics*, *9*, 1116–1131.

Aldón Mínguez, D., Ruiz Costa-Jussà, M., & Rodríguez Fonollosa, J. A. (2016). Neural machine translation using bitmap fonts. In *Proceedings of the eamt 2016 fifth workshop on hybrid approaches to translation (hytra)* (pp. 1–9).

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Balasubramanian, S., Jain, N., Jindal, G., Awasthi, A., & Sarawagi, S. (2020). What's in a name? are bert named entity representations just as good for any other name? *arXiv preprint arXiv:2007.06897*.

Belinkov, Y., & Bisk, Y. (2017). Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Bengio, Y., Ducharme, R., & Vincent, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems*, *13*.

Bibal, A., Cardon, R., Alfter, D., Wilkens, R., Wang, X., François, T., & Watrin, P. (2022). Is attention explanation? an introduction to the debate. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 3889–3900).

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. In *International conference on machine learning* (pp. 1613–1622).

Bostrom, K., & Durrett, G. (2020). Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720*.

Breiman, L. (1996a). Bagging predictors. *Machine learning*, *24*, 123–140.

Breiman, L. (1996b). Stacked regressions. *Machine learning*, *24*, 49–64.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... others (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, *30*.

Clark, J. H., Choi, E., Collins, M., Garrette, D., Kwiatkowski, T., Nikolaev, V., & Palomaki, J. (2020). Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, *8*, 454–470.

Clark, J. H., Garrette, D., Turc, I., & Wieting, J. (2022). Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, *10*, 73–91.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Dai, F. Z., & Cai, Z. (2017). Glyph-aware embedding of chinese characters. *arXiv preprint arXiv:1709.00028*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... others (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Eger, S., Şahin, G. G., Rücklé, A., Lee, J.-U., Schulz, C., Mesgar, M., ... Gurevych, I. (2019). Text processing like humans do: Visually attacking and shielding nlp systems. *arXiv preprint arXiv:1903.11508*.

Forbes, M., Holtzman, A., & Choi, Y. (2019). Do neural language representations learn physical commonsense? *arXiv preprint arXiv:1908.02899*.

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, *14*(771-780), 1612.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050–1059).

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., ... others (2023). A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 1–77.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *International conference on machine learning* (pp. 1243–1252).

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning* (pp. 1321–1330).

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 16000–16009).

Hendrycks, D., & Gimpel, K. (2016a). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Hendrycks, D., & Gimpel, K. (2016b). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Izsak, P., Berchansky, M., & Levy, O. (2021). How to train bert with an academic budget. *arXiv preprint arXiv:2104.07705*.

Jelinek, F. (1998). *Statistical methods for speech recognition*. MIT press.

Jin, D., Jin, Z., Zhou, J. T., & Szolovits, P. (2020). Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 8018–8025).

Jurafsky, D., & Martin, J. H. (2023). *Speech and language processing* (3rd ed.). Pearson.

Kawakami, K., Dyer, C., & Blunsom, P. (2017). Learning to create and reuse words in open-vocabulary neural language modeling. *arXiv preprint arXiv:1704.06986*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kong, L. (2019). Cyprien de masson d'autume, wang ling, lei yu, zihang dai, and dani yogatama. a mutual information maximization perspective of language representation learning. *arXiv preprint arXiv:1910.08350*, *5*.

Kudo, T., & Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, *30*.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Laves, M.-H., Ihler, S., Kortmann, K.-P., & Ortmaier, T. (2019). Well-calibrated model uncertainty with temperature scaling for dropout variational inference. *arXiv preprint arXiv:1909.13550*.

Li, J., Zhao, W. X., Nie, J.-Y., & Wen, J.-R. (2023). Renderdiffusion: Text generation as image generation. *arXiv preprint arXiv:2304.12519*.

Lin, Z., Trivedi, S., & Sun, J. (2023). Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 10012–10022).

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Mielke, S. J., Alyafeai, Z., Salesky, E., Raffel, C., Dey, M., Gallé, M., ... others (2021). Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*.

Mielke, S. J., & Eisner, J. (2019). Spell once, summon anywhere: A two-level open-vocabulary language model. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 6843–6850).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.

Mikolov, T., Sutskever, I., Deoras, A., Le, H.-S., Kombrink, S., & Cernocky, J. (2012). Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, *8*(67).

Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., ... Lucic, M. (2021). Revisiting the calibration of modern neural networks. *Advances in Neural Information Processing Systems*, *34*, 15682–15694.

Moosa, I. M., Akhter, M. E., & Habib, A. B. (2022). Does transliteration help multilingual language modeling? *arXiv preprint arXiv:2201.12501*.

Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 29).

Nguyen, D.-K., Assran, M., Jain, U., Oswald, M. R., Snoek, C. G., & Chen, X. (2024). An image is worth more than 16x16 patches: Exploring transformers on individual pixels. *arXiv preprint arXiv:2406.09415*.

Nivre, J., De Marneffe, M.-C., Ginter, F., Hajič, J., Manning, C. D., Pyysalo, S., ... Zeman, D. (2020). Universal dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643*.

OpenAI, R. (2023). Gpt-4 technical report. arxiv 2303.08774. *View in Article*, *2*(5).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, *21*(140), 1–67.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Rogers, A., Kovaleva, O., & Rumshisky, A. (2021). A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, *8*, 842–866.

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, *88*(8), 1270–1278.

Rust, P., Lotz, J. F., Bugliarello, E., Salesky, E., de Lhoneux, M., & Elliott, D. (2022). Language modelling with pixels. *arXiv preprint arXiv:2207.06991*.

Salesky, E., Etter, D., & Post, M. (2021). Robust open-vocabulary translation from visual text representations. *arXiv preprint arXiv:2104.08211*.

Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Skinner, B. F. (1957). *Verbal behavior*. New York: Appleton-Century-Crofts.

Spärck Jones, K. (2004). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, *60*(5), 493–502.

Sun, B., Yang, L., Dong, P., Zhang, W., Dong, J., & Young, C. (2018). Super characters: A conversion from sentiment classification to image classification. *arXiv preprint arXiv:1810.07653*.

Taylor, O. (2004). Pango, an open-source unicode text layout engine. In *Proceedings of 25th internationalization and unicode conference.*

Tenney, I., Das, D., & Pavlick, E. (2019). Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., ... others (2019). What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. In *International conference on machine learning* (pp. 10347–10357).

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., ... others (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vernikos, G., & Popescu-Belis, A. (2021). Subword mapping and anchoring across languages. *arXiv preprint arXiv:2109.04556*.

Vig, J. (2019). A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Wan, A. (2021). Fairness in representation for multilingual nlp: Insights from controlled experiments on conditional language modeling. In *International conference on learning representations.*

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wang, C. (2023). Calibration in deep learning: A survey of the state-of-the-art. *arXiv preprint arXiv:2308.01222*.

Wang, C., Balazs, J., Szarvas, G., Ernst, P., Poddar, L., & Danchenko, P. (2022). Calibrating imbalanced classifiers with focal loss: An empirical study. In *Proceedings of the 2022 conference on empirical methods in natural language processing: Industry track* (pp. 145–153).

Wettig, A., Gao, T., Zhong, Z., & Chen, D. (2022). Should you mask 15% in masked language modeling? *arXiv preprint arXiv:2202.08005*.

Wittgenstein, L. (1922). *Tractatus logico-philosophicus*. Routledge & Kegan Paul.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... others (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations* (pp. 38–45).

Wu, S., & Dredze, M. (2019). Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *arXiv preprint arXiv:1904.09077*.

Xiao, Y., Liang, P. P., Bhatt, U., Neiswanger, W., Salakhutdinov, R., & Morency, L.-P. (2022). Uncertainty quantification with pre-trained language models: A large-scale empirical analysis. *arXiv preprint arXiv:2210.04714*.

Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., & Hooi, B. (2023). Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... others (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the ieee international conference on computer vision* (pp. 19–27).