Parsing Heraldic Blazons

Alexandra Purcarea

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2023

Abstract

This paper presents a natural language processing approach to parsing blazons, a language which describes heraldic figures. Three main priorities were addressed: handling unknown words, identifying blazon components and their positions, and resolving blazon ambiguity. Using part-of-speech tagging, 641 previously unknown words were successfully assigned tags. Context-free grammar rules were implemented to build a hierarchical blazon structure that identified various blazon components. Blazon ambiguity was resolved by implementing an elimination system that tracked color dependencies and ruled out incorrect parses, resulting in an average decrease from 7.66 to 2.81 parses per blazon.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Alexandra Purcarea*)

Acknowledgements

Firstly, I would like to thank my supervisor, Julian Bradfield. Always understanding, he gave me a great deal of flexibility and allowed to take ownership of this project and shape it according to my vision. Our discussions on blazonry were always interesting and inspired me to improve my writing.

I am also grateful to my flatmates and friends, who really made this year great. Their encouragement and companionship helped me through some challenging times, and I will miss them dearly when I move to London.

Special thanks go to my boyfriend, Owen, for being my rock throughout this year. His kindness and assistance were invaluable, and his moral support helped me overcome some of the toughest obstacles.

Finally, I want to thank my family, including my beloved dog, Choco. Although they were far away, they always showed their love and curiosity for my work, and their encouragement was a source of motivation throughout this journey.

Table of Contents

Chapter 1

Introduction

Natural language processing (NLP) is a field that has greatly evolved in recent years, and its parsing methods are now used for many tasks, from text generation to machine translation and image captioning. A large part of the subject's success has been due to its deep understanding of language grammar and syntax. In the case of commonly spoken languages, there are numerous models which parse lexical categories of words and chart the dependencies between them. Understanding the structure and meaning of a sentence is aided by processes such as word sense disambiguation, which identify the sense of a word within its larger context.

Low-resource languages as well as constructed languages unfortunately do not have a similar amount of resources that understand their grammatical structures and specific vocabulary - while progress is being made with models such as multilingual BERT [9], they do not always perform ideally. Moreover, constructed languages in particular are lacking in documentation and pre-trained corpora that can be used for NLP tasks.

One such constructed language that has been widely used throughout European history is the language of blazonry. It refers to the description of heraldic figures, most notably coat of arms, and consists of highly specialised vocabulary and grammatical structure. Blazons are descriptions that have been traced to the 13th century and are still very much in use today, as many institutions (nations, army regiments, universities etc.) and families have their own coat of arms. However, this language illustrates the limited reach of natural language processing: as its grammar differs from English grammar to some extent, parsing it is not an intuitive or straightforward process.

This is where the project aims to improve the current state of natural language processing for the language of blazonry. While previous studies on this topic exist, they make use of bounded methods such as regular expressions, which limit their understanding of words that they have not documented themselves. Blazonry is an evolving language, as coats of arms have been using modern visual elements (e.g. planes), and therefore a hard-coded approach which handles each word manually can have shortcomings, in addition to it being time-consuming.

The primary aim of my dissertation is to parse blazons using natural language processing methods that allow me a greater deal of flexibility in parsing new words from the language. I divide this task into three main priorities which lend themselves to different sections of the project:

1. Handling unknown words. By using natural language processing methods, I do not have to manually label all words in blazons, which although fairly finite, are still numbered in the thousands. I aim to create a model which assigns a lexical tag to each word it encounters, even those that pre-trained models do not recognise.
2. Identifying blazon components and their positions. Blazons are representations of visual elements, which are divided into specific sections (such as the background of the shield, or its division), and therefore do not lend themselves well to common syntactical relationships such as subjects or predicates. I aim to create custom categories and build hierarchical structures for blazons.
3. Resolving blazon ambiguity. There may be different interpretations of a blazon in terms of structure, such as dependencies that are inferred in multiple ways. I aim to eliminate ambiguity in blazons by having processes in place that rule out incorrect parses.

1.1 Outline

Following the introduction, Chapter 2 discusses the background of blazonry, including its history, structure, previous works, and context within natural language processing. Chapter 3 discusses the design of the implementation, justifying choices that were made for the three main aspects of the implementation.

The first section of the implementation can be found in Chapter 4 and explores part-of-speech tagging for blazons, preceded by finding and pre-processing input data. It is followed by context-free grammar rules and chart parsing in Chapter 5. The final part of the implementation is disambiguation, which can be read in Chapter 6.

Finally, the conclusion in Chapter 7 provides a summary of the achievements of this project, its strengths and weaknesses, as well as discussion of future work.

Chapter 2

Background

2.1 An introduction to blazons

Heraldic blazons are formal written descriptions of heraldic figures, most importantly coats of arms and flags. The language employed in blazonry - the art of creating blazons - is ”concise, (...) definite, and explicit” and also has a very specific nomenclature ”of Norman-French origin” [5]. As such, blazons can be seen as a constructed language with highly specialised vocabulary, grammar, and syntax [14].

According to archaeologist Charles Boutell, heraldic language became systematised in the reign of Henry III (1216-1272), and continued spreading until the Tudor dynasty [5]. In the second half of the nineteenth century, heraldry saw a renewed interest and increase in documentation: many sources, such as Boutell’s Heraldry and Parker or Elvin’s heraldic dictionaries, originate from that time period. The rise in popularity of heraldry during the Victorian era may be explained by the Gothic Revival, a Romantic movement that encouraged self-expression through heraldry, historical re-enactments, and widespread medieval imagery in everyday objects and art. [21]

Blazonry remains significant to this day, being used to describe military heraldry, coats of arms of families or municipalities, as well as national emblems. In addition to it being preserved, new symbols have been added to blazonry that reflect change in society and how communities represent themselves. Figure 2.3 represents a common new symbol in heraldry - an airplane.

Blazons can range from very simple to very complex. Figure 2.1 shows a coat of arms with a blazon consisting of a single word: *Gules*, translating to red. In contrast, Figure 2.2 shows a coat of arms with a longer blazon, in which elements of different colours and positions are represented in the concise style of the language.

The following section outlines particularities of and useful terminology for the grammar, vocabulary, and structure of blazonry.

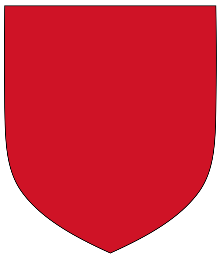


Figure 2.1: The coat of arms of the Albret family of Landes, France. Blazon: *Gules.*
[8]

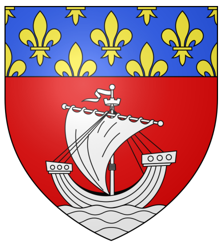


Figure 2.2: Coat of arms of Paris. Blazon: *Gules, on waves of the sea in base a ship in full sail Argent, a chief Azure semé-de-lis Or.*
[18]

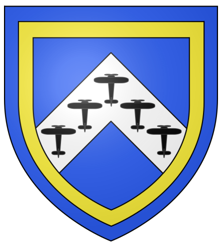


Figure 2.3: Coat of arms of the town of Orly in France.
[1]

2.2 Grammar, vocabulary, and structure

As previously mentioned, blazons consist of highly specialised terminology; both the names of components and the words used to describe them have been in use for over eight centuries, and many of them originate from Old French and Anglo-Norman [10]. Below is a concise introduction of common terminology that will be used throughout this dissertation, with examples of common vocabulary words for each component.

A blazon may contain one or more of the following:

- Field: the 'background' of the coat of arms.
- Divisions of the field: lines of partition that divide the field in two, three, or more sections. Examples: *Per pale, tierced-in-fess, quarterly.*
- Variations: patterns that cover the field, more complex than simple divisions. Examples: *Chequy, lozengy, semé.*
- Tinctures: chosen from a limited palette of colours and patterns (furs), they describe all of the elements of a blazon. Examples: *Or, azure, ermine.*
- Ordinaries: simple geometrical features running across the shield. Examples: *Chief, cross, saltire.*
- Charges: graphical features such as geometric shapes, animals, plants, or objects (weapons, tools, royal symbols). Examples: *Lion, fleur-de-lys, crown.*
- Attitudes: the position of a heraldic figure, usually an animal. In this project, we

will treat attitudes as both the direction that they are facing and the stance they are assuming. Examples: *Rampant, couchant, regardant.*

- Positions of charges: less well-defined than other categories, this describes words that mark the positions of charges in relation to one another, or in terms of their alignment on the field. Examples: *In bend, between, on.*

Appendix A provides examples of common instances of many of the elements described above.

It is important to note that variations are not as well-documented as other categories listed above - Boutell defines varied fields as ”surfaces [that] are usually tinctured of some one metal and some one colour alternating” [5]. However, the term ’variation’ seems to only appear in more modern texts, and will be used for practical reasons.

As a constructed and semi-structured language, blazons possess certain particular grammatical features and conventions [14]:

- As in the French language, colour descriptors come after the element they describe. *3 choughs sable* translates to 3 black choughs.
- Colours for elements are inferred. *A chief and 3 choughs sable* implies that both the chief and the 3 crows are black. The economy of the language is emphasised in this feature, which although not enforced, is often used. However, due to this, there may be slightly different blazons for the same coat of arms, leading to ambiguity when parsing them.
- There are rules of tincture to follow, mainly that ”metal shall not be placed upon metal, nor colour upon colour” [10]. Metals refer to *Argent* (silver) and *Or* (gold). For very complex blazons, these rules help disambiguate the position of elements such as charges.

The order of a blazon is generally the following: field (partitions/variations if any, then tincture), ordinaries (and any charges on or around them), then finally charges. Following the French grammatical style, a colour is given after the item they refer to.

It is important to note that in this dissertation, I will only discuss the escutcheon (shield) in a heraldic achievement; blazons can also describe other heraldic elements, such as mantles, supporters, or badges.

2.3 Existing studies

Formal academic studies that attempt to parse blazons are limited, while projects by heraldry enthusiasts are more widespread.

In 2021, Michael Mulder at the University of Twente created a compiler for blazons [17] that uses a method of semi-parsing called Iterative Lexical Analysis (ILA) in order to parse blazons into an intermediate representation, which is then used for graphical rendering. ILA encodes tokens as strings, then using pattern matching, it translates them into a generalised intermediate language until the entire blazon is parsed. An example of a translated blazon would be ”PARTITION COMMA COLOUR AND COLOUR”, represented in a parse tree where each token is matched with its respective text. The author acknowledges that grammatical inferences discussed in Section 2.2, such as describing two elements with a single colour, were not correctly interpreted by the regular expressions in his algorithm.

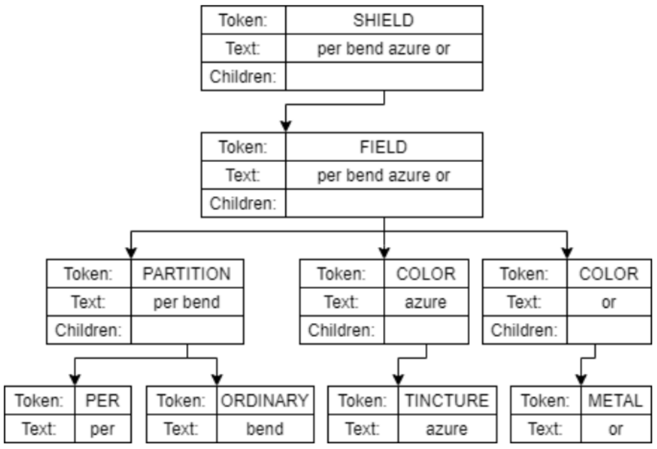


Figure 2.4: Intermediate representation of a blazon in Mulder’s study. [17]

Iterative Lexical Analysis was created by A. Cox and C. Clarke to better detect syntactical irregularities in code by repetitively using regular expression matching. A syntax tree is constructed from the bottom up by identifying smaller elements which then build up to larger ones, as seen in Figure 2.4. Clarke and Cox only test their method on error-free code, with high precision and recall, and recognise the need to ”improve its tolerance for irregular code” [7]. However, although it is a constructed, semi-structured language, blazonry is not as precise as a programming language. As previously mentioned, grammatical features and conventions are sometimes used, although not enforced, therefore leading to ambiguity. Therefore, using Iterative Lexical Analysis for longer, more ambiguous blazons could prove unsuitable.

While researching the topic, one of the most comprehensive resources has been the DrawShield website created by Karl Wilcox [23]. The website renders a shield from a blazon and covers a wide variety of divisions, edge types, charges, and other elements. The parser behind the website is open source and converts heraldic blazons to Abstract Syntax Trees - in addition, it uses the Levenstein algorithm for spelling corrections. In comparison to Mulder's project, Wilcox's parser correctly interprets colour inferences: *Gules, a chief and chevron or* correctly renders a red shield with a golden chief and golden chevron.

Abstract Syntax Trees are used as representations of the syntax of a blazon, similar to Figure 2.4. In his parser, Wilcox first applies a complex tokeniser on the blazon, then matches words to patterns using regular expressions, as seen in Mulder's algorithm. The reason for the correct interpretation of colour inferences is the function lookAhead() that checks the colour modifiers of the next graphical element in case the current element lacks one. The downside of using regular expressions for pattern matching is that, save for known irregularities such as colour inferences, the algorithm will not know how to classify unknown elements or partitions. Wilcox's codebase contains thousands of examples of charges, colours, and other elements, with new ones being constantly added. However, manually classifying a non-exhaustive list of blazon elements is time-consuming, and can produce a biased notation.

2.4 NLP in the context of blazonry

2.4.1 Part-of-speech tagging methods

Part-of-speech (POS) tagging is a process that assigns lexical categories to tokenized words in a sentence, and is an important component of many NLP projects such as speech synthesis, analysis of a corpus' word usage, or predicting behaviours of unknown words [4].

In this task, the Viterbi algorithm is often used to calculate POS tags for the entire sentence by calculating the most likely sequence of tags [20]. As seen in Figure 2.5, the algorithm starts from the beginning of the sentence and calculates probabilities by multiplying the observation likelihood (the probability that it is a certain tag given the word) and the transition likelihood (the probability that it is a certain tag given the previous tag). After the probabilities are calculated across the sentence, the best path is calculated backwards using backpointers in order to maximise the total probability.

The advantages of the Viterbi algorithm are its efficiency, accuracy, and ease of use. Pre-trained corpora allow the algorithm to calculate highly accurate observation and transition likelihoods, creating a versatile model that takes context into consideration and looks at the sentence as a whole when assigning tags. On the other hand, the algorithm has certain setbacks - for one, it only looks backwards for context, possibly missing out on additional meaning in the sentence. Moreover, unless smoothing is used, the observation likelihood of an unknown word will be 0 across all lexical categories.

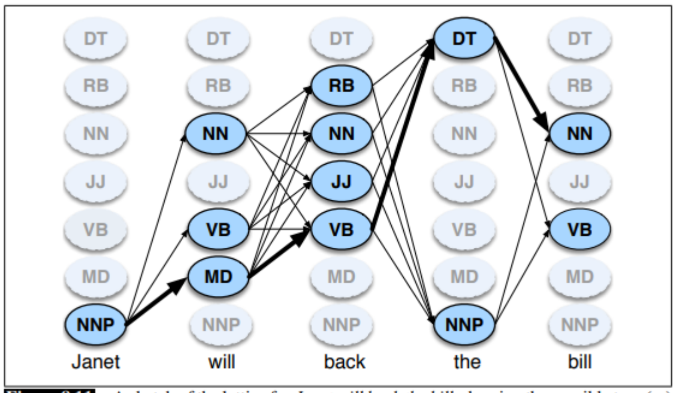


Figure 8.11 A sketch of the lattice for *Janet will back the bill*, showing the possible tags (q_i) for each word and highlighting the path corresponding to the correct tag sequence through the hidden states. States (parts of speech) which have a zero probability of generating a particular word according to the B matrix (such as the probability that a determiner DT will be realized as *Janet*) are greyed out.

Figure 2.5: Viterbi lattice representation.
[12]

A model that has been shown to be effective in various language processing tasks is BERT (Bidirectional Encoder Representations from Transformers), whose creators intended for it to be beneficial even for "low-resource" tasks [9]. BERT builds on top of transformer architecture by adding multiple layers of transformer blocks and bidirectional self-attention. Transformer blocks consist of encoder and decoder stacks - an encoder maps a sequence of word vectors to feature vectors of smaller dimension, whereas a decoder does the opposite. Encoding can also be seen as generating a contextual representation of all words in a sentence.

The encoder and decoder are connected through an attention mechanism, which combats the drawbacks of the encoder-decoder model by directing focus to a small set of words that convey the most information in a sentence. The attention model also adapts its context vector to each time step, in order to predict the next word. In the first transformer model proposed by Vaswani et al., attention is defined as:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

given the set of queries Q, keys K, and values V (all matrices).

Using self-attention has proved successful in ”a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations” [22].

As the model’s name suggests, it uses bi-directional pre-training, incorporating context from left-to-right as well as right-to-left directions, unlike OpenAI’s GPT model in which ”every token can only attend to context to its left” [9]. BERT has been shown to not only perform well in part-of-speech tagging, but also infer dependencies between said parts of speech within a sentence. In Jawahar et al.’s 2019 study, determiner-noun and subject-verb dependencies ”are captured accurately” [11]. As such, BERT can not only be used for word tagging, but it can also operate at the higher level of a sentence by detecting dependencies.

Alongside its versatility, BERT has the advantage of taking into account the entire context of the sentence and selecting words through its attention mechanism that may aid correct tagging. On the other hand, the model is so large that it might need significantly more data for training than a simpler one such as Viterbi in order to achieve similar success rates.

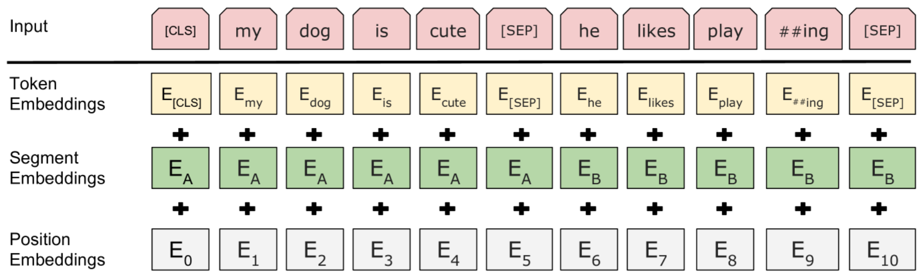

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.6: BERT Input Representation.
[9]

2.4.2 Context-free grammar

Context-free grammars (CFG) were formally defined by Noam Chomsky in 1956 as ”phrase-structure grammar” [6] and refer to a logical series of rules that could describe any phrase in a given formal language. They are defined by a finite vocabulary, a finite set of ”initial strings” (often called terminals), and a finite set of rules. All rules can be written as:

$$X \rightarrow Y$$

where X and Y are strings that belong to the vocabulary. Y can be written as a series of variables, terminals, or a combination thereof. All rules for X may be written on the same line, separated by vertical lines. Rules may be recursive, in the sense that a variable could refer to itself, allowing for any number of strings to follow it such as in this example:

$$X \rightarrow X\ Y$$

meaning that X might be followed by an endless number of Y instances. The final goal is to match the entire sentence to an end node often denoted as S.

Several efficient parsing algorithms exist that determine whether a string can be generated from CFG rules and if so, what that parse might look like. The algorithm we will be using in this project is chart parsing, which uses well-formed substring tables (WFST) [4]. A WFST essentially consists of a triangular matrix that records the positions of words on its axes. Consider the grammar rule:

$$A \rightarrow B\ C$$

Each substring cell in the matrix is evaluated and populated with nonterminal A, if the substring can be split in such a way that it matches B and C. The parse continues upwards through the matrix, and is considered complete and successful if the entire string is matched to a full sentence S.

Context-free grammars have the advantage of creating a strict, hierarchical structure of language on as many levels as needed. They provide a convenient tree structure which is simple to understand visually but which also contains dense information about each sentence. Nevertheless, it does not come without some drawbacks - the accuracy of a parse highly depends on the rules to be written strictly, so as not to allow any ambiguity. In addition, CFGs are not sensitive to how frequent grammatical structures are, and judge all possible parses to be equal.

Chapter 3

Design

As seen in Mulder's and Wilcox's projects, there are still areas for improvement when it comes to parsing blazons. Considering their achievements as well as their shortcomings, three priorities arise in order to successfully parse various blazons: handling unknown words, identifying blazon components and their positions, and resolving blazon ambiguity.

Parsing heraldic blazons is a task that requires multiple steps, each with their data processing, experiments, results and interpretation. The following chapters will each treat an individual step in detail. The overall goal of the project, as previously stated, is to parse heraldic blazons and provide a logical representation of their language. That description alone allows for some degree of interpretation and personal decision-making. Knowing that the techniques for processing and interpreting data were up to my discretion, I will provide justification for my decisions, which took into account the information at my disposal at the time.

Sections 2.1 and 2.2 background chapter provide information on the context and structure of blazons, from which we conclude that while blazons are constructed using highly specific methods and terminology, the size of their vocabulary is not small, nor is it fixed. As new words are being added to blazons (words for charges, in particular), a parsing tool needs to learn how to deal with unknown words. Moreover, writing an exhaustive list of manual labels for each word would be incredibly time-consuming and arguably redundant, given the accuracy of machine learning models. For all of the reasons stated above, it was natural that the first step of the project was part-of-speech (POS) tagging for each word in a blazon. This matches our first priority of the project: handling unknown words is made possible using probabilistic tagging models.

Once we have a part-of-speech label for every word, it becomes easier to derive hierarchical structure from the blazon. A simple example would be a charge followed by its colour - while part-of-speech tagging might point that there is a noun followed by an adjective, a tree structure would show the relationship between the two words, as well as their location in a coat of arms. I achieve a tree representation for each blazon by creating my own context-free grammar (CFG) rules, starting from rules involving individual terminals up to the large components of a blazon (e.g. a field followed by a charge). The custom CFG rules derive significant meaning and allow for a great deal of flexibility in parsing blazons on many levels. The second requirement of the project, which is identifying blazon components and their positions, is therefore met.

Finally, we want to consider the third priority of our project: resolving blazon ambiguity. This section was designed after the results of the previous CFG outputs, which pointed to a large number of parse trees caused by an inevitably ambiguous logical language for blazons. In the context of ambiguous language, clarifying blazon representations meant reducing the number of parses returned for each blazons, and more specifically removing objectively incorrect parses. The introduction of Chapter 6 explains the use of filters for disambiguation - rather than creating a function that chooses a singular correct parse tree, I created one that selectively removes blazons that do not obey a specific set of rules. This difference in approach means that I can write more generalised rules about dependencies in blazons, and also accept that sometimes there may be more than one correct parse - or that, at least, the ambiguity of the blazon cannot be resolved without more context from the blazon's writer.

The final result of the project will be parse trees returned by a blazon that has been tokenized into separate words with POS tags. Tree structures convey a significant amount of information - their leaves can be labelled with names of blazon components and sub-components, down to terminals, and their hierarchy can be seen from the intuitive visual representation.

This project was written in Python, within a singular Jupyter notebook that contains all steps of the implementation. A virtual environment was used to keep track of all libraries used, which are imported at the top of the file. While some cells in the notebook are used for computing large structures (such as parsing all blazons into CFG trees), others serve the purpose of illustrating examples of various parts of the implementation.

Chapter 4

Part-of-speech Tagging

4.1 Introduction

The first section of the project aims to obtain and pre-process blazon data, then implement part-of-speech (POS) tagging for each separate word in our blazon list. After finding a blazon database, the data needs to be converted into a list of separate blazons, which in turn will be tokenized sentence by sentence in order to calculate POS tags. Tokenization can be achieved in multiple ways, and will be discussed further in the implementation section.

Part-of-speech tagging is done with a preset list of tags: in our case, the Penn Treebank tagset, containing 36 distinct tags [2]. These tags are returned by models that are pre-trained on the English language. Table 4.1 below shows examples of tags that I will discuss in this section.

Tag	Description
CC	Coordinating conjunction
DT	Determiner
IN	Preposition/subordinating conjunction
JJ	Adjective
NN	Noun, singular
NNP	Proper noun, singular
VB	Verb, base form

Table 4.1: Commonly occurring Penn Treebank POS tags.

4.2 Datasets and pre-processing

Finding a reliable blazon database proved to be one of the more challenging aspects of the project. While many resources on blazonry exist, they are often in the form of scanned books that lack easily searchable lists of blazons to parse. Additionally, works such as A. de Sainte-Marie's *Histoire généalogique de la Maison Royale de France* contain not only blazons of noble families, but also lengthy information about family trees and other text which would be difficult to remove in order to extract only the blazon [8]. Historically, blazons were often included in the context of a noble family's history, genealogy, or participation in wars, which makes it challenging to find blazons in isolation.

Fortunately, a newer source solved the problem. The *Ordinary and Memorial* of the Society for Creative Anachronism's search engine provides access to 64,547 blazons. Accessing the database for private use is allowed under "fair use" copyright laws [3]. The blazons from the website are generated by users, making it an unofficial corpus; however, the SCA has its own blazon guidelines that draw from Boutell, Fox-Davies, and other established references on the subject [16].

To extract the blazons, a query for all blazons was performed using the term ".", and the first 1000 results were scraped as HTML code from the website. The page's code was then processed to separate individual blazons and remove all HTML tags. Blazons that began with the string "(Fieldless)" or "(Tinctureless)" were excluded from the corpus, the justification being that historically blazons would always have a field, making this a modern addition. This filtering reduced the number of blazons by 129.

4.3 Methods and implementation

BERT was initially used to implement part-of-speech tagging for blazons. The case-sensitive version of multilingual BERT was chosen for tokenization and tagging, which is pretrained on the Penn TreeBank and achieves an F1-score of 96.69 [19].

The first attempt at tokenization made some issues apparent. Consider the example of the blazon ”Argent, a square weaver’s tablet purpure.” Its tokenization resulted in the following sequence:

> [’A’, ’##rgent’, ’,’, ’a’, ’square’, ’we’, ’##aver’, ”’”, ’s’, ’tablet’, ’pu’, ’##rp’, ’##ure’]

In this tokenized blazon, the words argent, weaver, and purpure are incorrectly split, and the latter is even divided into three tokens. The only correct separation is the possessive pronoun following ”weaver”, which is a feature that occurs in around 5% of blazon inputs. As all blazons contained similar words being incorrectly separated, POS tagging would clearly suffer from inaccuracies. This is partially caused by BERT not recognising French-origin words such as ”argent”, and therefore not knowing their root. The question arises whether tokenization should try to find the roots of words at the expense of excessively splitting them until they become unrecognizable.

The solution to this problem was to add all words to the tokenizer’s vocabulary, leading to no word splitting at all. A total of 641 new words were added, including words like ”argent” but also derivative forms of a known word root, such as ”lilies”. Derivative forms included plurals, possessive pronouns, past participles, and hyphenated words. While the lack of separation in derived words may be detrimental to POS tagging, the inclusion of 300-400 French-origin words in the vocabulary is arguably more significant towards a POS tagger that understands words in their context.

In summary, tokenization in our case consisted of splitting words from a sentence, though not separating any plural, possessive, or hyphenated forms. The comma was kept in blazons, but not the full stop at the end.

After finalizing the tokenization process, the multilingual, cased BERT model and its tokenizer were utilized as parameters in a classification pipeline. The pipeline then produced part-of-speech tags for each word in a given sentence, along with corresponding confidence scores for each token. BERT returns tags for one sentence at a time, leading us to look at how individual blazons were tagged.

The results of BERT’s POS tagging, which will be discussed in the next section, prompted new experiments using a different model. NLTK’s recommended part-of-speech tagger uses the Viterbi algorithm trained on Hidden Markov Models to calculate the most likely sequence of tags [15]. This model tokenizes by splitting contractions (e.g. *they’ll*), treat commas as a separate token, and remove end-of-line periods. In practice, though, tokenization is the same as in the BERT model, as blazons do not use pronouns and will therefore not have contractions. Moreover, this tokenizer does not separate possessive forms, making it identical to the previous one. After tokenization, NLTK’s pre-trained *pos_tag* function is called, returning tagged blazons. The results will be compared and contrasted between the BERT and NLTK models.

4.4 Results and evaluation

Both the BERT and NLTK model return part-of-speech tags for individual blazons; moreover, the NLTK model does not return confidence levels for its tagging. For these reasons, presenting results is more difficult to do on a larger level across the entire data.

The table below compares the part-of-speech tags of the BERT model compared to the NLTK model - it can be seen as anecdotal evidence for larger issues with the models.

Token	BERT tag	NLTK tag
Or	CC	CC
,	,	,
in	IN	IN
fess	NN	JJ
two	CD	CD
lettuces	NN	NNS
vert	NN	VBP

Table 4.2: Blazon example tagged by two models.

In terms of inaccuracies, two distinct types can be identified: one is model-specific, the other is general inaccuracy caused by the distinctive language of blazons. The clearest example of the latter would be the word ”or”, which in blazonry represents the colour yellow, but which in modern English is a coordinating conjunction. A model-specific inaccuracy related to the BERT model is that it assigns the adjective tag to far fewer words than the NLTK model, in cases where the words are in fact adjectives. In general, the distribution of POS tags differs across the two models, as illustrated in the figure below.

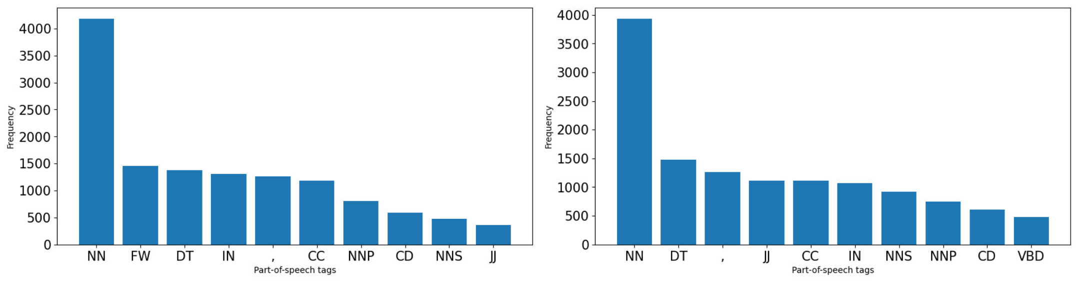


Figure 4.1: The 10 most frequent POS tags for BERT (left) and NLTK (right) models.

Figure 4.1 illustrates significant differences in part-of-speech tagging between the BERT and NLTK models. BERT’s second most frequent tag is FW (foreign word), whereas it does not figure at all in NLTK’s top 10 tags. In addition, it tags fewer determiners with the DT tag, and correctly identifies less than half the number of adjectives compared to the NLTK model. In contrast, the NLTK model features a verb tag (VBD) in its top 10 list, and has a more equal spread between different tags following the NN category. As will be seen in the interpretation section, these differences informed the final choice between the two models.

BERT tag	NLTK tag	Count
FW	NN	660
NN	JJ	597
NNP	NN	438
NN	NNS	431
IN	NNP	256
FW	JJ	239
NN	VBD	237
JJ	NN	161
FW	DT	106
NN	NNP	105

Table 4.3: Most common occurences of tag differences between the two models.

Building on previous results, I wanted to more discretely track differences between POS tags between the two models, in order to understand what gets 'mistagged' by the two models. Table 4.3 illustrates the high frequency of BERT assigning the FW tag where the NLTK model assigns nouns, adjectives, or determiners. Moreover, BERT can misunderstand adjectives, plural nouns, verbs, and proper nouns and assign the NN tag to many of them.

Not only do the tags differ significantly, but so do the runtimes of the two models. BERT initially requires a *TokenClassificationPipeline* object to be initialised, after which tokenized blazons can be passed as inputs to the pipeline. The first tagged line therefore requires significantly more time to compute, at 1.6 seconds. After the pipeline is established, it can be used for every blazon, reducing the runtime to 0.6 seconds per blazon. However, NLTK does not require anything to be initialised before running its *pos_tag* function, which has an average runtime of 0.2 seconds per blazon. These differences compound across the selected corpus of nearly 900 tokenized sentences and amount to a difference of approximately 6 minutes when tagging all blazons. The relative slowness of the BERT model prompted me to explore other tagging options, hence why I also experimented with NLTK.

4.5 Discussion

Overall, part-of-speech tagging solves the pressing issue of how to handle unknown or unexpected words in blazon - by assigning a tag to every word, it becomes easier to handle words and create rules for blazons. As opposed to hundreds or thousands of words to parse, we now have a finite list of 36 tags that builds our initial, low-level structure of a blazon.

However, as was seen in the results section, not every tag is assigned correctly. Adjectives in particular pose a problem to the NLTK, and especially to the BERT model. As adjectives follow the noun in blazonry, models trained on English would more reluctantly assign a JJ tag after a NN one. This was a compromise that had to be made, since training BERT or any other model on French grammar would solve the word order problem, but would not understand most words as they are of English origin. Initially in this part of the experiment, incorrect tags (such as 'or' almost always tagged with CC) were kept the same - however, in the next chapter's experiments, they had to be changed in order to correctly create a higher-level parse for blazons. Unfortunately, evaluating the accuracy of tagging adjectives would require manually checking hundreds of blazons, as there are no pre-tagged blazons to test on.

Adding a new part-of-speech tagging method proved to be instrumental to the project - while the BERT model offered confidence scores for each of its tags, the NLTK model had more nuance in its tagging and gave only 4 words the FW (foreign word) tag. While foreign words may be important to detect in other natural language tasks, in our case most blazon words could be considered foreign as they are not in modern English language dictionaries, making this tag significantly less meaningful. The most important difference in terms of tags was the number of adjectives, as they are the second most important category after nouns in a blazon - a model that distinguishes better between nouns and adjectives can then describe a charge and its colour in a blazon more easily. Considering the better distribution of tags in the NLTK model, as well as its shorter runtime, I made the decision to only use that model.

The difference in performance between BERT and NLTK is not intuitive, as they were both trained on the same Penn TreeBank corpus, and use the same part-of-speech tagset. However, Section 2.4.1 might provide insight into the problem - we established that BERT may need a larger dataset in order to return results as good as a Viterbi algorithm. The lacking additional data, in combination with the large amount of unknown words, may have caused its attention model to underperform. This may explain the large amount of words labelled FW, as the model may have needed more context and more pre-training before understanding how to handle the specific language of blazonry. NLTK, on the other hand, uses a Viterbi algorithm which relies on transition and observation probabilities alone to infer tags. This means that even if it does not intuitively understand what a word is, it follows the rules of English grammar, e.g. it knows that a noun would follow a preposition.

Chapter 5

Building Blazon Structure with Context-free Grammar

5.1 Introduction

Given the part-of-speech tags extracted from blazons in the previous chapter, we have the building blocks necessary to build up a hierarchical structure of a blazon using context-free grammar (CFG). As the structure of a blazon is highly specific, typical CFGs that deal with the grammar of a sentence are impractical for our purpose, calling for custom rules to be created. The rules that will be implemented are on several levels: the lowest-level ones will label terminals or combinations of terminals, while the highest-level ones will complete the construction of an entire blazon from recognizable components such as fields, divisions, and charges.

As an end result, we want to see complex blazons being parsed correctly from top to bottom, using chart parsing and returning a tree structure. Moreover, we want the parser to correctly identify dependencies and relationships between different components of the blazons. This section's experiments show an iterative process of adding rules, starting with simple ones and building up to complex, recursive patterns. As this part of the project is implemented manually, evaluating the CFG itself is potentially difficult - it is easier, however, to judge the experiments' accuracy based on the output (trees) across the entire corpus.

5.2 Implementation

The implementation of the context-free grammar rules was a gradual process, starting with rules for the simplest type of blazons: ones where the field (consisting of a tincture) is followed by a simple charge (a determiner followed by a noun). At the beginning, words such as colours were quite loosely defined as multiple parts of speech, such as proper nouns, adjectives, or coordinating conjunctions.

As the blazon structures became more complex, these less strict definitions led to incorrectly parsed blazons. This prompted me to rethink the tags around words that belonged to 'fixed' categories such as divisions and tinctures. For instance, tinctures have a fairly short set of words, with no unexpected additions to their category. On that account, I thought it suitable to manually change the word tags of some words in order to assign them to stricter categories.

Blazon component (tag)	Examples
Tincture (JJ)	Or, ermine
Charge (NN)	Chief
Variation (VAR)	Chequy, lozengy
Division (DIV)	Party, quarterly
Attitude (ATT)	Sejant, passant
Position (POS)	In, between

Table 5.1: Manual overrides to fixed blazon categories.

As previously mentioned in Section 2.2, variations are a less well-defined term in blazonry; however, due to their distinct grammatical structure, they have been added as a component type for practical reasons.

Table 5.1 lists the blazon components whose words received newly created tags, with the only exception being tinctures who were changed into adjectives - this change had two main benefits. Firstly, words that were consistently labelled wrongly, such as 'chief' as an adjective instead of a noun, could now be parsed into the structure correctly. This had previously caused problems with words like 'Or' which was given the same tag (CC) as 'and', but which needed to be parsed as a colour instead of a conjunction in between two charges or tinctures, and oftentimes led to no parse tree at all. Secondly, the addition of new categories separate frequently occurring specialised words from the usual nouns and adjectives, aiding the process of distinguishing a higher number of blazon categories. For instance, without custom tags, *per fess* and *in fess* would have to be parsed as the same category, even through the former is a division and the latter is the positioning of charges. Appendix B contains a full list of the manually overwritten words.

Once the additional categories were added, it became simpler to create higher-level rules for elements such as the variation of a field or the attitude of a charge. In some cases, rules are made to be recursive, like with charges where a great number of words can follow them in no particular order. This creates the need for relative flexibility in blazon construction, and writing patterns that allow for an unspecified number of elements describing a main one. Instead of creating highly specific rules about how many words can follow a charge, we can instead define CFG rules such as:

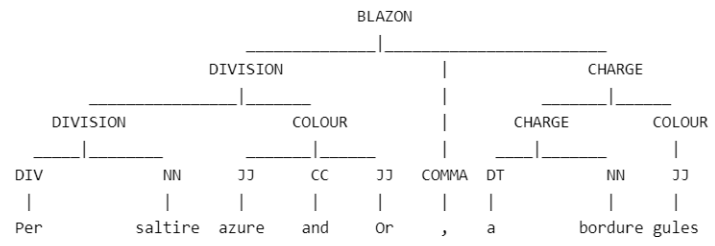
$$\text{CHARGE} \rightarrow \text{NUMBER CHARGE}$$

Appendix B contains a list of all CFG rules that were used to build tree structures for blazons. The rules are parsed from the bottom up, making the order crucial to a correct parse, and begin with defining the terminals. As we are building up the tree structure from tags rather than specific words, tags are seen by the *ChartParser* algorithm as the 'words' in this scenario. These rules are parsed in a strict order, from left to right on a line, and from bottom to top in order to build hierarchical structure. For this reason, and for recursion to work, I occasionally found the need to define components on multiple levels. On the bottom line, there would be more terminals, while the top would contain higher-level patterns as well as rules that I wanted to enforce at the very end of parsing for a specific category. For instance, knowing that the main components of a division are a division name and its colours, I added that rule above the rules that define divisions using terminals such as DIV, NN, or CD.

The *blazon_tree()* function links each blazon word to its tag, and allows it to be printed under the parse tree generated by the context-free grammar. This was a fairly complicated workaround to implement, and it relies on deconstructing the parse tree, manually removing the part-of-speech tag to its corresponding word, and then reconstructing the tree. Moreover, the function returns all possible trees returned by the parser, in the case where the grammar rules allow for multiple interpretations, but only graphically displays the first one. Using the *pretty_print()* function from NLTK shows a readable hierarchical structure of a blazon.

5.3 Results and evaluation

Figure 5.1 illustrates how blazons were processed and returned as parse trees with the help of CFG rules. In terms of qualitative evaluation, it is important to note that not only is correct labelling of components important, but also their position in hierarchy in relation to other elements. For instance, in the figure, we want the words 'azure' and 'Or' to be on the same level, as they are joined by a coordinating conjunction. Moreover, the determiner should immediately join the charge following it, which I have ensured by adding the rule first when defining charges.

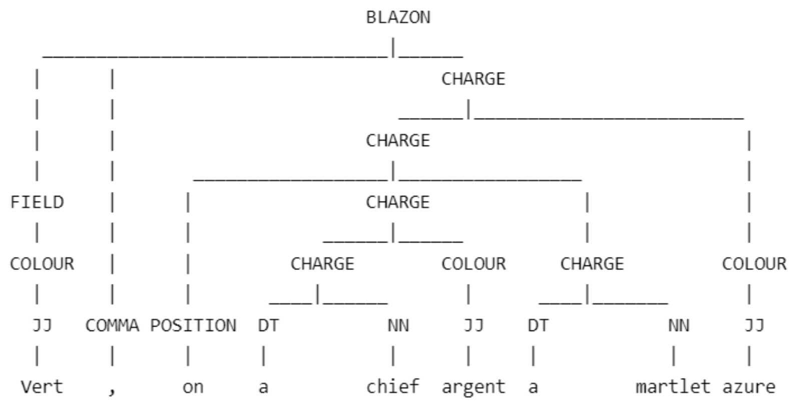


```
                                 BLAZON
                    _____________|_____________________
               DIVISION                    |             CHARGE
      _______________|______               |          ______|_____
 DIVISION               COLOUR             |     CHARGE          COLOUR
 ____|_______        ______|_____          |    ___|______          |
DIV         NN     JJ      CC     JJ     COMMA DT          NN       JJ
 |          |      |       |      |        |   |           |        |
Per      saltire azure    and     Or       ,   a        bordure   gules
```

Figure 5.1: Tree Representation of a Blazon using CFG.

When it comes to quantitative evaluation, we can assess the success of the CFG grammar based on their coverage, as well as on the number of parses generated for each blazon. Coverage is calculated as the number of blazons that are parsed without errors by the CFG function compared to those with errors, and initially in the experiments it was a very unequal 6/865 split. Currently, the coverage stands at 285/586, having significantly improved partly due to the introduction of the new tags. Each additional rule also led to the number of successful blazons growing at an increasingly faster rate, as their various permutations would account for more types of blazons.

The other evaluation method calculates the average number of parses returned by the grammar for each blazon. This number stands at a surprisingly high 7.66 parses per blazon, meaning that there could be around 8 trees returned for a blazon that the grammar considers to be 'correct'. Upon closer inspection, while some trees are equally 'correctly' parsed (the differences being arbitrary, in what follows a charge first for instance), others are clearly incorrect but still technically follow the grammar rules. Figure 5.2 illustrates an example for the latter scenario - it assigns the colour 'azure' to both the chief and the martlet, even though the chief already has argent as its assigned colour. The other parse tree of this blazon shows the correct dependency between the last charge and its colour.



```
                                 BLAZON
   ________________________________|_______
  |      |                              CHARGE
  |      |                          ______|__________________________
  |      |                       CHARGE                              |
  |      |       __________________|_________________                |
FIELD    |      |                CHARGE              |               |
  |      |      |            ______|______           |               |
COLOUR   |      |         CHARGE        COLOUR     CHARGE         COLOUR
  |      |      |       _____|______      |      _____|_______       |
  JJ   COMMA POSITION  DT          NN     JJ    DT           NN     JJ
  |      |      |      |           |      |     |            |       |
 Vert    ,      on     a         chief  argent  a         martlet azure
```

Figure 5.2: Example of an incorrect tree parse.

5.4 Discussion

Manually adding tags, or changing already existing ones, may arguably be counterproductive to the part-of-speech tagging we have seen previously; ideally, we would have a highly accurate tagger that could identify old Anglo-Norman words, but seeing as to how that is unlikely, there was a need to manually change a select few tags. Moreover, tagging is most useful for identifying previously unseen words, which would likely be more obscure charges or adjectives describing them - manually overriding the tags for the most frequent words seen in blazons does not make the previous section of the implementation futile. In fact, POS tagging greatly helped this part of the implementation, as more blazons could be parsed than if I had had to manually tag each word myself.

There are many advantages to creating context-free grammar for blazons - the parse trees seen in the two figures convey a great deal of information about a blazon's components, and how they further break up into smaller sections down to the word level. The tree structure also allow us to understand the different levels and where different words interact with each other - as previously mentioned, we want to make sure that words that are of equal meaning should be on the same hierarchical level, and similarly with larger sections of a blazon.

On the other hand, individually defining custom context-free grammar rules leaves room for bias. My process of writing rules was based on rules of blazonry from well-established sources, starting with defining the colour of the field. However, the process also required iterative development, and refining rules had to be based on individual blazons. This might add more error to parsing - consider the case of trying to fit the parser around one unusual blazon by adding a less-than-strict rule which then could generate too many trees for other blazons (some of which will be incorrect). Moreover, I had to make compromises between creating highly specific rules and trying to fit wrongly tagged words into the picture. In the case of colour, I added proper nouns (NNP) as a possibility, though if I'd had better tagging, that would not have been necessary.

The number of trees returned by each blazon was higher than I had conjectured, and led to further thinking about what disambiguation might look like for blazons. It is clear that the more parse trees there are for a blazon, the less meaning could be extracted from it and the more ambiguous the structure becomes. For this reason, selectively ruling out incorrect parses may be the way forward for disambiguation of blazons.

Chapter 6

Disambiguation for Blazons

6.1 Introduction

Klint and Visser define an ambiguous context-free grammar as ”a language in which some sentences have multiple interpretations” [13]. While previous works propose methods that specify the correct parse for a sentence, they introduce a ”framework of filters” defined by rules that eliminate parse trees, instead of selecting them.

In a similar manner, I considered that disambiguating blazons made more sense in terms of elimination rather than selection - the latter would arguably be more arbitrary, as some of the parse trees are equally valid. As such, disambiguation would focus less on the idea of ’correctness’ - which is difficult to check externally for the specific language of blazons - and more on ruling out parses with incorrect dependencies. This idea could take several forms, though not all of them are equally effective. Adding CFG rules before parsing has the potential to multiply parse trees, but changing their order may prioritise a correct parse first. Removing CFG rules may decrease the number of tree parses, but in doing so, would also decrease the number of blazons that are parsed at all.

In the end, I decided on adding elimination rules for parse trees after generating all tree parses. Doing so would keep the *blazon_tree* function intact, and would take its trees as the input, returning a list of the trees that passed the elimination rules. Similarly to the context-free grammar section, writing the rules was an iterative process - after writing an elimination pattern, I would check the statistics returned by the disambiguation function and move on to another rule by looking at blazons with many parse trees. I chose the output to remain a list of trees, as I have found they are the most ’readable’ format and pack substantial information about a blazon’s structure.

6.2 Implementation

Figure 6.1 shows an example of one parse tree out of two returned by the *blazon_tree* function for the blazon *Azure, a fleur-de-lys argent and a bordure gules*. In this example, the parse is incorrect, as it assigns the colour *gules* to both the fleur-de-lys and the bordure, even though the former is assigned a colour.

```
(BLAZON
  (FIELD (COLOUR (JJ JJ)))
  (COMMA ,)
  (CHARGE
    (CHARGE
      (CHARGE (CHARGE (DT DT) (NN NN)) (COLOUR (JJ JJ)))
      (CC CC)
      (CHARGE (DT DT) (NN NN)))
    (COLOUR (JJ JJ))))
```

Figure 6.1: Text representation of a parse tree.

I implemented the *disambiguate_trees* function to traverse the lines of the text, which are the leaves of the parse tree, and to determine whether criteria for a correct tree are met given previous lines. The first two rules were:

- If a charge is followed by a colour on the same line, the charge on the next line should also be followed by a colour.
- If given a position, the charge afterwards should be assigned a colour.

These rules were chosen based on analysing a series of blazons and identifying commonly occurring inaccuracies across the entire corpus. The overarching issue present in many parse trees had to do with colour dependencies - for instance, a charge being assigned two or more colours, or none at all. However, the first two elimination filters did not catch all of the faulty trees, especially in the case of longer, more complex blazons. An additional rule had to be introduced in order to solve the issue adequately:

- If a charge is followed by a colour on the same line, the next charge regardless of position should be assigned a colour. If a charge is not followed by a colour on the same line, the next colour will be assigned to it.

This rule introduces a long-range dependency rule for colours that tracks the most recent charge in the blazon and decides on the validity of the tree based on whether the following charge has a colour or not. Unlike the previous two rules, this rule does not follow specific lines, but instead looks for the next charge in a tree.

All three rules are implemented with the use of booleans as flags (which keep track of whether a colour rule is met or not) which change as they iterate through the lines of the tree. Each line is checked for 5 conditions, which include the presence of a charge, a position, and the presence or absence of a colour. The flags change accordingly on each line based on these checks, and trees are ruled out if a rule is broken. For instance, one if statement checks if a line does not contain a charge but has a colour - if the previous charge was already assigned a colour, this is not a valid tree, and will not be added to the list of trees returned by the function.

6.3 Results and evaluation

Table 6.1 charts the average number of trees returned for each blazon in the corpus (of the blazons that were parsed successfully by the CFG function). Initially, we start at 7.66 parses, which decreases by over 2 parses once the first two rules are implemented, and which sees an even more significant change once the long-term dependency rule is introduced. Our final disambiguation implementation scales down the number of parses to 2.81 per blazon. With these three rules implemented, 42 out of 295 blazons are reduced to zero tree representations, meaning that the function ruled out all their possible parses.

Implementation step	Average parses per blazon
Before implementing rules	7.66
After short-range colour and position rules	5.24
After long-range colour dependency rule	2.81

Table 6.1: Average parses per blazon at various steps of the implementation.

As blazon lengths vary significantly, it may also be useful to look at the average parses per blazon across blazon lengths. Figure 6.2 shows the average parses before (orange) and after disambiguation (blue). The number of parses is shown on a logarithmic scale in order to better visualise small and large numbers in the same chart. As blazons get longer and more complex in terms of clauses, disambiguation rules down more parse trees, as can be seen by the increased gap between averages starting at around a length of 18 words in a blazon.

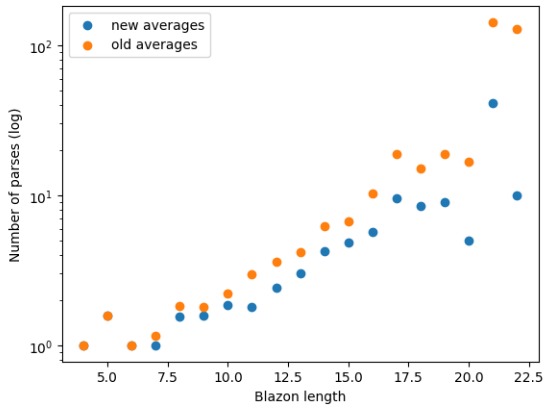


Figure 6.2: Average number of parses per blazon length, before and after disambiguation.

Of the remaining parses for each blazon, many can be considered to be 'technically correct' - for instance, as charges are defined so broadly, there may be more than one correct interpretation of a blazon, where dependencies within a charge may happen at different levels but still be valid interpretations. Evaluating this, however, would need to be done manually, and introduces a level of bias. In the blazon *Purpure, a camel rampant and a mount Or*, the colour *Or* could arguably belong to just the mount or the camel as well as the mount. The disambiguation algorithm does not eliminate either parse, leaving it up to individual interpretation.

6.4 Discussion

When I initially implemented the disambiguation rules, I got an average number of under 2 parses per blazon - however, that came at the cost of success rate, as 112 blazons had 0 parse trees after disambiguation. This was a clear sign that there were trade-offs to make between the strictness of disambiguation rules and the number of successful parses. In the end, getting to below 3 parses per blazon was a significant improvement compared to the beginning of implementation.

Evaluating the correctness of this algorithm proves difficult, as previously stated. We might evaluate parses as correct or not, but are disambiguation failures necessarily a fault of the function, or do they reveal underlying problems with a blazon? On one hand, we want to rule out as many incorrect parse trees as possible and still keep correct ones; on the other hand, the blazons themselves could be written in such a way that they don't obey typical rules of blazonry. *Azure, on a heart Or a brown bear statant proper* is an example of a blazon where the colour *brown* precedes its noun, rather than following it as per usual blazonry guidelines. It could be seen as an adjective-noun compound noun, although that would mean we cannot technically infer the colour of the bear. Disambiguation fails in this instance, as it does not identify a colour following a charge. The question then arises whether we should allow rules to accept such blazons, but that would come at the expense of disambiguation being much less effective.

Overall, this section of the implementation is arguably the most 'hardcoded' one, as I had to manually define highly specific rules about what strings should be on a line for the tree to count as a 'valid' parse. Moreover, we mostly looked at whether colours follow charges, whether that was on the same line or on a longer range, but more rules could potentially be implemented to disambiguate trees. In the case of divisions, we could track the number of colours that should follow them; we could also track variations as they occur in a charge, to ensure that they are not followed by colours.

Chapter 7

Conclusion

Chapter 1 states that the primary purpose of this project was to parse blazons using NLP methods. Moreover, three main priorities arose when defining the task. I will present my achievements in relation to these three goals, and subsequently evaluate my findings.

1. Handling unknown words. Along all other tokens, 641 words that were previously unknown to the corpus were assigned part-of-speech tags.
2. Identifying blazon components and their positions. Main blazons such as tinctures, charges, variations etc. were introduced as non-terminals in a context-free grammar. Using them, I built a hierarchical blazon structure that successfully parsed one third of the blazon corpus.
3. Resolving blazon ambiguity. The high number of CFG parses returned for each blazon required filtering, which was implemented using flags that tracked colour dependencies across the blazon, and ruled trees out if the dependencies did not make sense. From an average of 7.66 parses per blazon, we saw a decrease to 2.81.

To sum up, all three priorities of the project were met using natural-language processing methods, and with demonstrable success.

7.1 Strengths and weaknesses

Some of the strengths of this project lie in the successful combination of NLP methods with manual functions or overrides, whenever necessary. As a constructed language, blazonry lies somewhere between a finite logical representation and natural language, and a hybrid language may require hybrid methods. There were certain areas where NLP did not contribute enough insight into the problem - for instance, POS tagging for specific words like ”or” - and I had to discern where and to what extent I should overwrite or rewrite parsing.

In Chapter 4, I had to make a significant decision that determined the direction of the entire project - switching from the BERT model to NLTK for part-of-speech tagging. Although studies reinforced BERT’s accuracy and ability to track dependencies [11], in practice it did not perform as well as I had expected it to, and I had to reorient towards a different model. This, in my opinion, demonstrates flexibility and an ability to adapt my methods based on results.

In addition, some tasks required to be implemented without relying on a previously existing model. In Chapter 5, I was constrained by pre-existing CFG rules that were modelled on the English language and had to write my own hierarchical rules from scratch, which was a lengthy iterative process that required a lot of trial and error. Although sources such as Boutell’s Heraldry mention the general order of elements in a blazon, it is by no means an exhaustive resource, nor does it consider all possible combinations of elements in the algorithmic manner that I require.

In terms of weaknesses, many initial struggles were caused by the sparsity of resources on blazons. While numerous older books exist on the topic, they are not indexed and do not contain easily parseable lists of blazons that I could use as input. The same lack of sources made CFG parsing more difficult to implement, as previously mentioned.

Another area for improvement throughout the project would be the evaluation methods in each section. Due to the lack of pre-trained corpora of blazons, it was difficult to find quantitative methods for evaluation, as accuracy can only be calculated when there is a ’correct’ set of worked examples for the task that we can compare our results to. Many quantitative measurements track whether a method is ’successful’ at parsing rather than ’accurate’. If I were to calculate precision or accuracy, I would have to manually annotate tags with correct POS tags, for example, which would have inbuilt bias. Moreover, it would be impossible to manually evaluate hundreds of parsed blazons, so these measurements would have to be calculated on a smaller subset of the input data, introducing more potential problems.

7.2 Future work

Moving forward, there are several directions for future work that one can pursue. Firstly, after achieving successful results from using CFG rules to identify blazon components and their relationships, it would be interesting to explore dependency parsing as an alternative approach. This method might allow for more accurate complex blazons and a more robust recursion system, though it requires hand-written dependency formalisms.

Secondly, incorporating graphical elements in blazon parsing would mean that a user can input a blazon and receive a drawn shield in return, similar to the function of the DrawShield website. Creating a front-end framework to display blazons would make the parsing easier to understand than previously, and it would provide an intuitive and user-friendly experience. Moreover, it is easier to visually understand the relationships between components than it is to read a parse tree, particularly in the case of a long blazon.

Finally, building on the previous point, future work can improve on the model's ability to infer the position of charges within the shield. While the position within a larger component is marked using CFG rules, we can move beyond this to build an intuitive understanding of the locations using key words (e.g. words such as 'sinister' or 'in chief').

Bibliography

[1] *Histoire et patrimoine*. Ville d'Orly.

[2] Penn treebank P.O.S. tags. *University of Pennsylvania School of Arts and Sciences*, 2003.

[3] Search forms for the SCA armorial, Dec 2022. `https://oanda.sca.org/`.

[4] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with python*. O'Reilly, 2009.

[5] Charles Boutell. *Heraldry, historical and popular*. R. Bentley, 3 edition, 1864.

[6] N. Chomsky. Three models for the description of language. *IEEE Transactions on Information Theory*, 2(3):113–124, Sep 1956.

[7] A. Cox and C. Clarke. Syntactic approximation using iterative lexical analysis. In *11th IEEE International Workshop on Program Comprehension, 2003.*, pages 154–163, 2003.

[8] A. de Sainte-Marie. *Histoire généalogique et chronologique de la Maison Royale de France, des pairs, des grands officiers de la Couronne & de la Maison du Roy & des anciens barons du Royaume*, page 207. Compagnie des libraires associez, 1730.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[10] Arthur Charles Fox-Davies and Graham Johnston. *A complete guide to heraldry*. T.C. & E.C. Jack, 1925.

[11] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, July 2019.

[12] Dan Jurafsky and James H. Martin. *Speech and language processing*. Pearson Prentice Hall, 2014.

[13] Paul Klint and Eelco Visser. Using filters for the disambiguation of context-free grammars. April 1997.

[14] R. Lee Humphreys. From blazonry to policespeak. *English Today*, 7(3):37–39, 1991.

[15] Edward Loper, Steven Bird, and Ewan Klein. *5. Natural Language Processing with Python*.

[16] Bruce Miller. A grammar of blazonry, Jul 2018. `http://heraldry.sca.org/armory/bruce.html`.

[17] Michael Mulder. Creating a Compiler for the Semi-Structured Language of Blazons, January 2021.

[18] City of Paris. *"Fluctuat nec mergitur", l'histoire de la devise de Paris*. Nov 2020.

[19] Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Firoj Alam, Abdul Rafae Khan, and Jia Xu. Analyzing encoded concepts in transformer language models. In *North American Chapter of the Association of Computational Linguistics: Human Language Technologies (NAACL)*, NAACL '22, Seattle, 2022.

[20] Luz Abril Torres-Méndez. The Viterbi algorithm. *McGill University Centre for Intelligent Machines*.

[21] Jack Turton. The social and cultural significance of Victorian heraldry. *The Victorian Web*, Aug 2017.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[23] Karl Wilcox. DrawShield. `https://drawshield.net/`.

Appendix A

Examples of blazon elements

All pictures are sourced from *Heraldry, an introduction*, an educational resource on the DrawShield website [23].

A.1 Tinctures

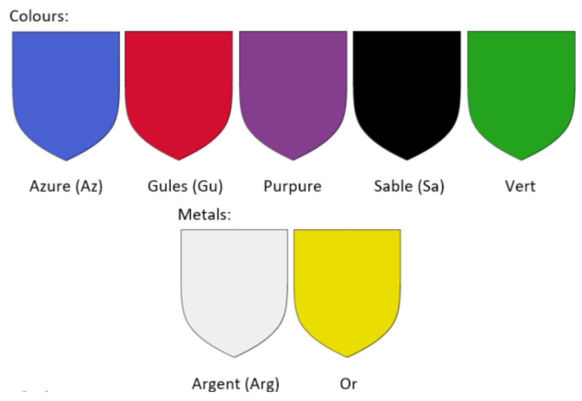


Figure A.1: Common colours and metals.

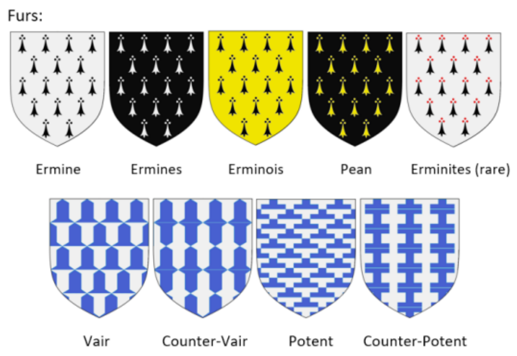


Figure A.2: Common tinctures.

A.2 Divisions and variations of the field

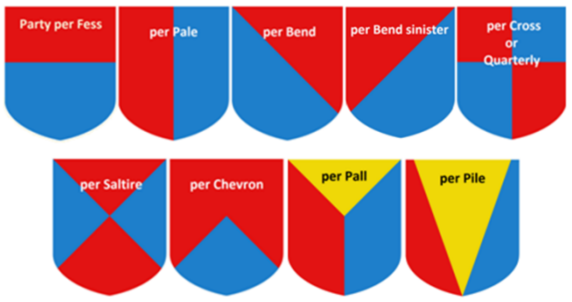


Figure A.3: Common divisions.

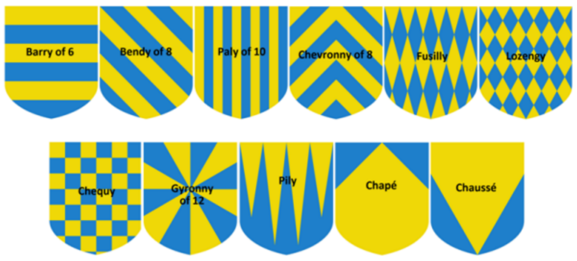


Figure A.4: Common variations of the field.

A.3 Charges

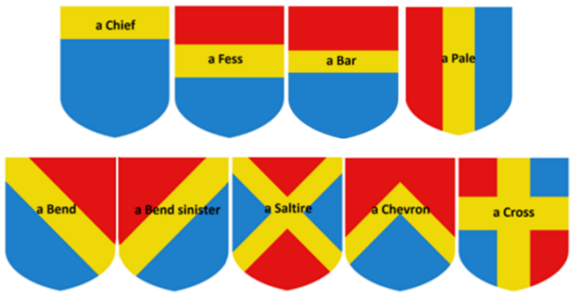


Figure A.5: Common ordinaries.

Appendix B

Full list of context-free grammar rules and select terminals

BLAZON → FIELD
→ FIELD COMMA CHARGE
→ FIELD COMMA CHARGE COMMA CHARGE
→ FIELD VARIATION
→ DIVISION
→ DIVISION CHARGE
→ DIVISION COMMA CHARGE
→ VARIATION COLOUR
→ VARIATION CHARGE
→ VARIATION COMMA CHARGE

FIELD → COLOUR

VARIATION → VARIATION COLOUR
→ COLOUR VARIATION

VARIATION → VAR
→ VAR IN CHARGE
→ VAR IN NNS
→ VAR VAR
→ VAR NN

DIVISION → DIVISION COLOUR

DIVISION → DIV NN
→ DIV IN CD
→ DIV

CHARGE → POSITION CHARGE

CHARGE → CHARGE POSITION COLOUR
→ CHARGE POSITION CHARGE
→ POSITION CHARGE CHARGE

→ POSITION CHARGE
→ CHARGE NN
→ CHARGE NNS
→ CHARGE COLOUR
→ NUMBER CHARGE
→ CHARGE VERB
→ CHARGE CC CHARGE
→ CHARGE ATTITUDE
→ CHARGE VAR

CHARGE → DT NN
→ DT NNS
→ DT JJ NN
→ NUMBER NN
→ NUMBER NNS
→ NN COLOUR
→ NNS COLOUR

POSITION → POSITION NN
→ DT POSITION IN

POSITION → POS

COLOUR → NNP
→ JJ
→ JJ CC JJ

VERB → RB VERB

VERB → VBD
→ VBN
→ VBZ

NUMBER → CD

B.1 Words with manually defined tags

New tag (abbreviation)	Manually selected words
Noun (NN)	Chief, bend, fess, fleur-de-lys
Adjective (JJ)	Or, gules, vert, sable, azure, argent, purpure, ermine, erminois, pean
Variation (VAR)	Gyronny, semy, wavy, annuletty, lozengy, semy-de-lys, goutty, increscenty, checky
Division (DIV)	Per, party, quarterly
Attitude (ATT)	Sejant, rampant, passant, couchant, dormant, statant
Position (POS)	In, within, to, on, between, orle