FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Vojtěch Lanz

Unsupervised segmentation of Gregorian chant melodies for exploring chant modality

Institute of Formal and Applied Linguistics

Supervisor of the master thesis:	Mgr. Jan Hajič, Ph.D.
Study programme:	Computer Science
Study branch:	Language Technologies and Computational Linguistics

Prague 2023

I would like to thank my supervisor Jan Hajič for his invaluable guidance, inspiration, and supervision throughout the project. His insightful recommendations and advice, especially in musicology and statistics, have greatly enriched my understanding. I truly appreciate the time he devoted to leading this work.

Title: Unsupervised segmentation of Gregorian chant melodies for exploring chant modality

Author: Vojtěch Lanz

Institute: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Jan Hajič, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Gregorian chant, as an oral musical tradition, was performed by singers that had to memorize thousands of melodies. Each melody has a set of properties, one of which is what mode it belongs to within the modal system. To understand the learning process principles of chants, it may be helpful to decompose melodies into smaller units and analyze their relationship to modality. In this work, we compare Bayesian and neural network unsupervised segmentation methods. We measure their performance on evaluation metrics we design in order to examine the chant's properties with respect to the memorization challenge considering the modality aspects. For this purpose, we have two datasets, one with over thirteen thousand antiphons and the other with over seven thousand responsories. We find the Pitman-Yor process to be a more fitting model than BERT for this particular task, especially the conditional Pitman-Yor process model we proposed to segment each mode independently. We provide several clear arguments that modality and chant segmentation are closely connected. We also dispute the claim by Cornelissen et al. [2020b] that the natural segmentation by chant words or syllables is best in terms of mode classification, and we provide a new state-of-the-art performance on the mode classification task.

Keywords: digital musicology, Bayesian methods, Pitman-Yor process, unsupervised segmentation, Gregorian chant

Contents

Introduction

Gregorian chant is a body of monophonic and unaccompanied vocal music consisting of only text and melody. Gregorian Chant has been the preferred music of the Roman Catholic church since the Early Middle Ages up until today and has also diversified into concert performances and recordings in the later 20th century.

Why do we discuss chant melodies together with statistical methods? Since Gregorian chants were originally an oral tradition, singers had to memorize thousands of melodies. At least until the broader acceptance of staff notation in the 2nd half of the 11th century, only one person, the "cantor", would likely have a written approximate sketch of the melody[1] available during a performance in liturgy, if at all. Memorizing numerous melodies prompts the question of how to remember them effectively. From this perspective, it makes sense to examine Gregorian chant using statistical methods.

The sketches, and later precisely notated melodies, were gathered in manuscripts such as graduals or antiphonaries. Each parish, monastery, or other ecclesiastical institution would be expected to have these liturgical books, and many of them have survived to this day. However, besides these books used for performance, we know of a different type of book: the tonary. In tonaries, the repertoire was structured not in order of performance but according to the relationships between the melodies themselves, thus offering a window into how the repertoire was structured to promote better recall. It raises the question of whether there was any system of melodic units describing the specific chant that helped to memorize all chants for the whole liturgical year. Certain types of chants are even limited to a small number of possible melodies. Musicologists have also observed that specific melodic phrases are frequently repeated (Ferretti [1934], Helsen [2008], Levy [1970]). This compositional nature of melodies is also observed in monodic liturgical repertoires of other cultures (Nuttall et al. [2019]). The theory of *centonization* was developed, which describes the method of composition as the assembly of melodic units into a unified entirety, resulting in a final melody (Ferretti [1934]). However, it is still unclear to what extent established melodic units were used (Hiley [1993]) and, if so, whether particular rules were applied to compound them into the final musical composition (Treitler [1975]). Although Treitler [1975] criticized centonization theory as impractical, we should not ignore the possibility of the existence of frequently occurring melodic motifs, even though they may not cover the entire composition. Each occurrence of a frequently recurring melodic unit could be encoded into a single piece of information instead of memorizing all the tones contained within the motif. Each such melodic unit would thus reduce the amount of information the singer would need to memorize, which would facilitate the entire process. Therefore, we believe it is a valid avenue of research to search for such units, which would allow the melody to be compressed into fewer components needed for memorization. This problem could be described as a melody segmentation, where the melody parts that don't contain any of the frequent melodic motifs will be segmented as single-tone segments. In contrast, the frequent and strong melodic motifs will be segmented

[1]This system of notation was called *neumatic*.

as one component, including all motif notes. Considering memory efficiency as a guiding principle for segmentation quality leads to unsupervised methods.

How individual chants are arranged and linked by specific final and initial melody phrases is closely tied to chant modality. Each Gregorian piece belongs to one of eight modes, which is an analogy of today's major (Ionian) or minor (Aeonian) music scales. Medieval music theory elucidates the modality of chant by utilizing the melody's initial and final notes and ranges to classify the mode. The empirical concept was slightly different, and the given definitions were not always fulfilled. It was not until the 11th century that modal theory and practical concept were closely related and interconnected, as melodies were often composed to be consistent with theory, just as theory was adapted to empirical elements and observations from oral tradition. The empirical concept, for instance, shows that chants in books called tonaries were primarily organized according to modes, forming distinct melodic families. This modal categorization already provides a system for memorizing chant melodies. Therefore, the question arises as to whether each mode should have a separate set of frequent melodic units in case of their existence. It would then be easier to memorize familiar melodic motifs within each mode separately and be able to combine them correctly, which would also facilitate the memorization process. Furthermore, it should be then easier to classify the mode of a chant based on its familiar and frequent melodic motifs. The chants within one mode are more similar and share more characteristics with each other than with other chants from other modes. Therefore, it's essential to consider and analyze mode behavior while finding the potential best melody segmentation.

To date, there has been no research on unsupervised segmentation of Gregorian chants. It is not known how such a statistically motivated segmentation could aid in understanding chant modality, but it is an idea enticing for chant scholarship. Consequently, no known evaluation score can accurately measure the quality of chant segmentation. However, there are several similar segmentation problems that many researchers have already explored, and we will build upon them. In this work, we will mainly focus on Bayesian methods. Although neural networks can be more powerful, they usually require much more data for the training process. Furthermore, there is no straightforward way of computing predicted segmentation probability, which could be essential since there are no gold data on Gregorian chant segmentations. We only have data on unsegmented melodies of two types of chants - antiphons and responsories. Antiphons are shorter and more straightforward, while responsories are longer and more complex. We aim to design multiple evaluation metrics that describe the given problem from different perspectives. Some score functions and observations will be focused on modality. Others will be exploring our results more generally. As there is uncertainty in finding the correct melody segmentation, and there is no certainty that such segmentation even exists, our work focuses primarily on determining the properties of chants regarding our unsupervised approaches.

In this study, our main objective is to explore the existence of melody segmentation into smaller units. What would such a segmentation look like if it existed? Is Cornelissen et al. [2020b] right that the segmentation based on the chant's text is the optimal approach? Are the potential melody segments associated with the chant's mode? These are the specific questions that we will address in our research.

The structure of this work is as follows. In Chapter 1, we will provide a theoretical background that covers the history and musical aspects related to Gregorian chants. Then we will describe the statistical background in Chapter 2. Chapter 3 will give an overview of existing research on Gregorian chants and other unsupervised segmentation tasks that could be applied to this area. In Chapter 4, we will describe and analyze the datasets used in our work, including any modifications and filtration. Chapter 5 will define our evaluation metrics and explain their motivations. Chapter 6 will introduce various models we use for unsupervised segmentation. In Chapter 7, we will compare the segmentation results of these models with baselines. Finally, we will conclude by reviewing the work done and discussing possible future directions for research in this area.

1. Gregorian Chant

An indispensable part of Roman Catholic Church services was monodic music called Gregorian chant[1]. Thousands of chants are still preserved, although a large part has not survived at all. The music genre is called after Gregory the Great, pope active in the years 590-604, but his actual merits are unclear. On the other hand, there are still a lot of further details about Gregorian chants that are already known. Many of them are described by Hiley [2009], and we will summarize that work in this chapter, primarily focusing on information related to our task.

Gregorian chant is a type of liturgical chant sung during Christian services. The liturgy stands for a cycle of services and feasts, and it encompasses everything performed in both major types of liturgy: the Mass and the Divine Office. (In Latin Christian worship, the term Office is used in the context of Office hours; the detailed difference between Mass and Office will be shown in Section 1.1.) The purpose of chant within liturgy is to express sacred Latin texts ceremonially and formally. The way how soloists or choirs sang chants related to the sacred performance space. There were symbolic significances of most of the details in Christian worship. The entire ritual, encompassing all its details and symbolism, as well as the chants with their texts, held greater importance than the meaning of the texts themselves. The lyrics usually conveyed messages of praise, gratitude, supplication for God's mercy, or recollection of significant events in the history of salvation. The whole cycle of services was quite complicated, especially since the Roman Emperor Constantine the Great (324-37) declared Christianity as the state religion. The services took place every day of the year. Each day featured various services with unique structures and content. But not all days were equally important. For instance, Sundays were the most important days during the week. Also, days related to Christ's life were more important than others. Days of commemorations of holy men and women were also more significant. But the complete cycle of services also includes most of the regular days and parts of their nights.

The position of a chant in liturgy dictates to which *genre* it belongs: antiphon, responsory, introit, gradual, etc. Each genre implies a certain form and style, especially in terms of complexity. Some genres of Masses or Offices have highly ornamented, perhaps virtuosic melodies. Then the understanding of the chant's lyrics is secondary. On the other hand, there are parts of services intoning a text on a single note with some occasional modifications - usually at the ends of sentences or verses. This pertains mainly to readings from Bible, prayers, or psalms. In between these extremes, there are antiphons, one of the Office chant genres, that are more complex than the single-tone chants, but its melodies are still not that complicated, so the text is entirely understandable.

Particular texts don't have to be sung with always the same melody. The way they are interpreted depends on the liturgical context. Similarly, melodies can be utilized for more than just one text. Furthermore, some of the melody phrases are characteristic parts of particular chant genres or of the modes that each chant is composed in. Some melodic units are therefore repeated more often

[1]`https://www.youtube.com/watch?v=WkjgycIb2J4&ab_channel=TopClassicalMusic`

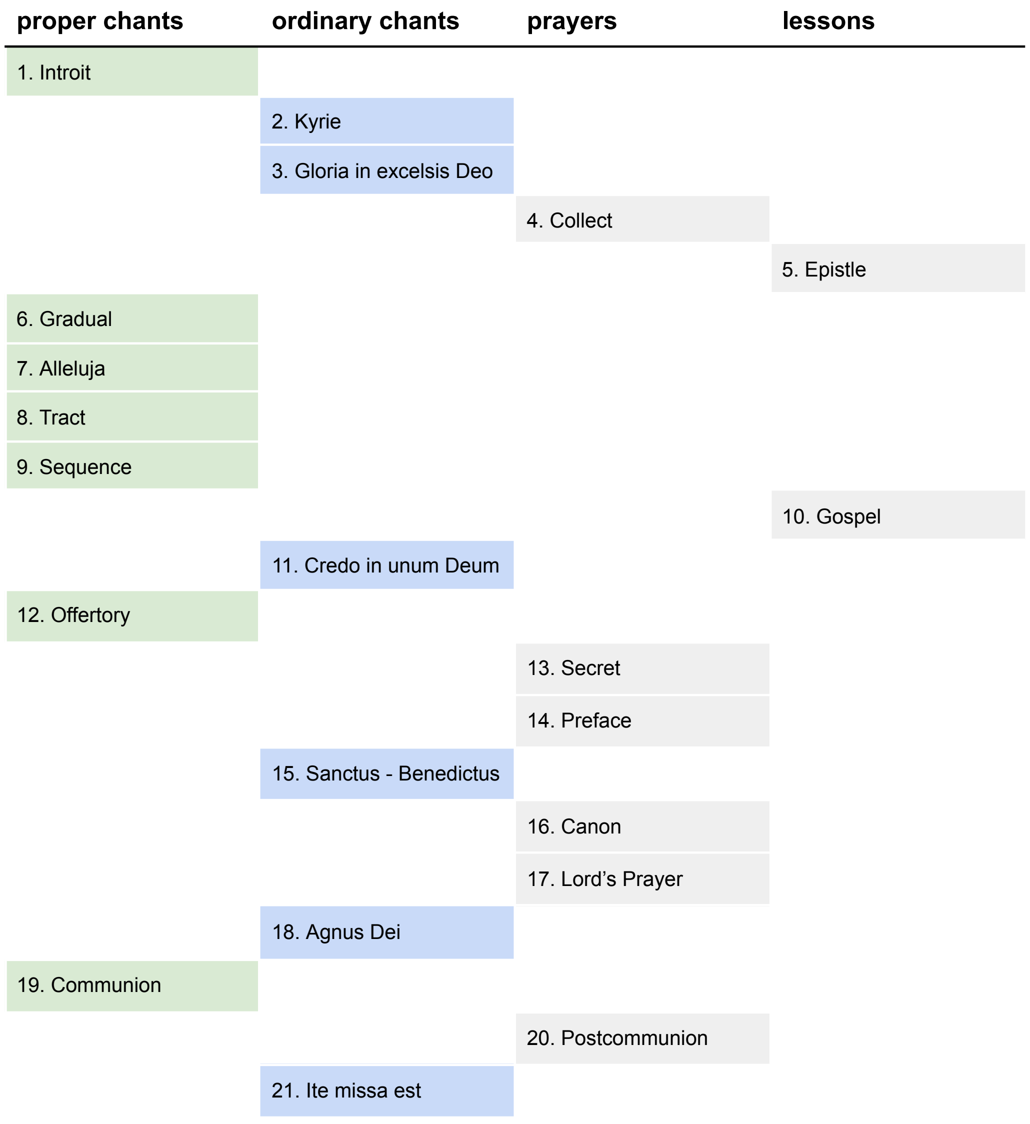
than others. The mode of a chant can, from the music-theoretical perspective, be loosely likened to the major and minor tonalities, but it is not all that tonality is. Section 1.2 describes the modality in more detail. There are also other aspects of compositional technique. It has been observed that in many melodies, typified melodic phrases are used. Many chants were formed within a framework that followed the structure of the text. But also chants followed the melodic movement norms. We will look at the Chant structure in Section 1.3. Nonetheless, these elements represent merely fragments of the overall composition, while a large part of the compositional technique is still unexplained. Discovering and describing the method would help to explain how singers learned, memorized, and performed thousands of melodies. But what's more, it would also be crucial to understanding thousands of melodies and how these melodies were transmitted across generations, especially with inexact notation during the early Middle Ages, as we will discuss in Section 1.4.

1.1 Mass and Office

Every day, worship took place filled with chants, following the cycle of the Liturgical year. The content of worship was affected by seasons, periods, and special holidays commemorating important events or significant individuals. Services were performed most of the day and extended into the night. A significant aspect of these services encompassed prayers and lessons, which are readings from sacred texts such as the Bible. Additionally, chants of Offices and Masses were part of services as well. Mass served primarily as an expression of gratitude, comprising distinct and unique components. On the other hand, the chants of hours of the Divine Office were more contemplative and exhibited notable similarities among them. Although there were some local variations, the daily service cycle's order of offices and masses was generally consistent across different times and locations.

1.1.1 Mass

The Mass, being the most common form of worship, serves primarily as an expression of gratitude and a profound encounter with the risen Christ, commemorating His institution of the Last Supper. Within the structure of the Mass, various elements such as prayers, lessons, and chants are incorporated. Some chants, namely Kyrie, Gloria, Credo in unum Deum, Sanctus, Benedictus, Agnus Dei, and Ite missa est, maintain consistent texts across different days. These are called the *ordinary* of mass. Then, there are chants with different prescribed texts for each day of the liturgical year, collectively called the *proper* of mass. The mass begins with the introit chant, which text varies depending on the liturgical context, symbolizing the entrance of the priest and their assistants into the church, proceeding towards the altar. Other chant genres with adaptable text, such as the gradual, alleluia, tract, and sequence, are often performed together. But it is not strictly mandated. Another chant genre without a fixed text is an offertory. Its role in the Mass is to begin the sequence of the performances leading up to the communion, wherein devout followers partake in the consumption of bread and wine. This communion is the most significant aspect and essential part of the Mass.



proper chants	ordinary chants	prayers	lessons
1. Introit			
	2. Kyrie		
	3. Gloria in excelsis Deo		
		4. Collect	
			5. Epistle
6. Gradual			
7. Alleluja			
8. Tract			
9. Sequence			
			10. Gospel
	11. Credo in unum Deum		
12. Offertory			
		13. Secret	
		14. Preface	
	15. Sanctus - Benedictus		
		16. Canon	
		17. Lord’s Prayer	
	18. Agnus Dei		
19. Communion			
		20. Postcommunion	
	21. Ite missa est		

Figure 1.1: Table of Mass parts and their order. Both chant types are colored. Proper chants are green, while ordinary chants are highlighted in blue color. (Hiley [2009])

1.1.2 Office

The Office, also called the liturgy of hours, consists of a daily cycle of eight services. The initial Office, known as Vespers, starts the daily service cycle before nightfall. (Traditionally, the day starts at sunset, not at midnight.) Vespers is followed by Compline. The Night Office, which is the longest of the hours, takes place during the night. Before drawn, there is a Lauds office, the most significant hour of the day. Subsequently, additional office hours Prime, Terce, Sext, and None, followed.

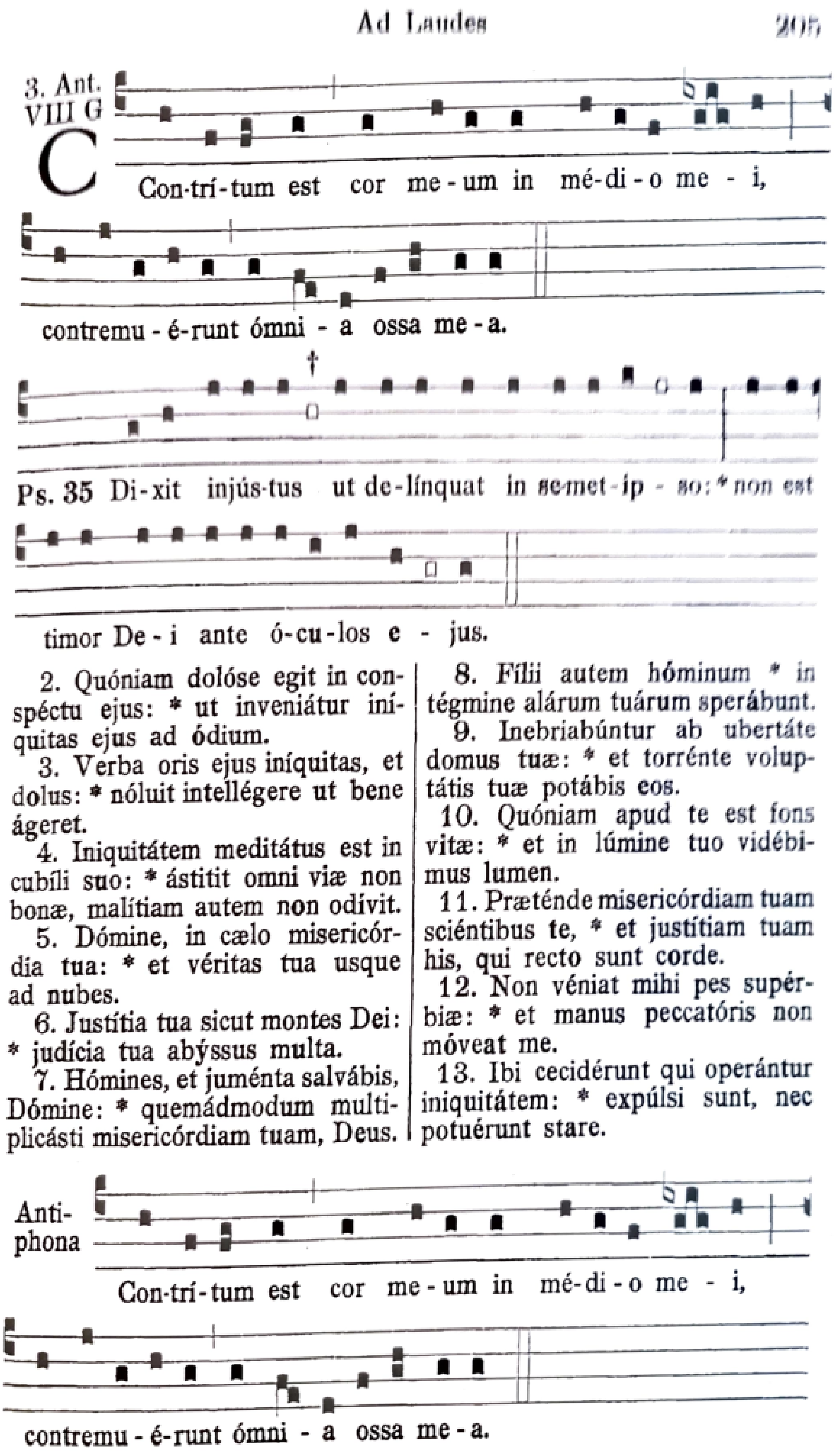
The Office, similar to the Mass, encompasses chants and prayers. Lessons are part only of the Night Office. The Office comprises various chant genres, such as antiphons, responsories, and hymns. These possess musical and melodic allure, particularly the grand responsories of the Night Office. But still, the psalm - chant performed similarly to lessons - holds a prominent position among the distinctive genres in the Office. Psalms are intertwined with antiphons in a structured manner: the antiphon is sung, followed by the performance of several psalms, and finally, the repetition of the initial antiphon. In later times, as musical notations became more prevalent, psalms were frequently indicated simply as a *differentia* at the end of the antiphon's score sheet. The differentia represents the concluding melody of the psalm, providing precise instructions on how the entire psalm should be performed since it is not a complicated chant genre. The exact number of psalms sung between antiphons is not strictly defined. It depends on the day's significance, location, or specific time. Additionally, even the entire antiphon-psalm-antiphon structure is usually repeated multiple times.

For each day, many antiphons and psalms were required. On a typical day, a minimum of twenty antiphons were prescribed. The number of responsories was lower, with at least three or four short responsories and an equal number of great responsories. The process of assigning specific pieces to particular days was quite complex. Antiphons, short responsories, and hymns were grouped into collections associated with each weekday. In addition, 150 psalms from the Book of Psalms were divided into individual days so that all of them could be sung each week. The situation became more complex regarding the great responsories of the Night Office. Sets of great responsories were exclusively designated for Sundays and feast days, while ordinary weekdays often repeat the great responsories from the Sunday set. In total, there were approximately 140 distinct antiphons, 30 different great responsories, and eight separate short responsories for the week. However, each special occasion brought a unique set of musical pieces, replacing the corresponding items from the regular weekday set, which was a common occurrence. Overall, there was a repertoire of over 2000 antiphons and over 800 responsories for the whole liturgical year, which the singers – in this case, all clergy – had to memorize. (And let's recall that these numbers don't even include the hundreds of chants for the Mass.)

1.2 Modality

In the early ninth century, the concept of modes in the context of Gregorian chant, which are categorizations of pieces preceding, by several hundred years, the major and minor scales of today, began to be practically used and explained in various historical records. However, there were significant differences, as eight modes were used instead of two. A more apt comparison would be with modes used today, and not only in jazz music, where they serve as a framework for building melodies and improvising over various chords and harmonic progressions. These encompass seven scales, all sharing the same number of sharps or flats, i.e., tones. The only variation lies in the permutation of the scale, which means the starting note is always distinct. There are other musical cultures using more strict modal systems, such as Indian ragas, Turkish maqam, or the Indonesian gamelan.

In Gregorian chants, the modes function in a similar manner. There are eight modes, all defined in medieval theory as octave-size selections on what we know as white keys on a keyboard. The medieval definition was expressed in terms of a final tone (d, e, f, or g) and range in relation to the final (either an octave up or a fifth up and a fourth down). This was a heritage of the tetrachord-based music theory of antiquity, where four tones filling a perfect fourth were the building


Ad Laudes 205

3. Ant. VIII G

Con-trí-tum est cor me-um in mé-di-o me - i, contremu-é-runt ómni - a ossa me-a.

Ps. 35 Di-xit injús-tus ut de-línquat in semet-ip - so: * non est timor De-i ante ó-cu-los e - jus.

2. Quóniam dolóse egit in conspéctu ejus: * ut inveniátur iníquitas ejus ad ódium.
3. Verba oris ejus iníquitas, et dolus: * nóluit intellégere ut bene ágeret.
4. Iniquitátem meditátus est in cubíli suo: * ástitit omni viæ non bonæ, malítiam autem non odívit.
5. Dómine, in cælo misericórdia tua: * et véritas tua usque ad nubes.
6. Justítia tua sicut montes Dei: * judícia tua abýssus multa.
7. Hómines, et juménta salvábis, Dómine: * quemádmodum multiplicásti misericórdiam tuam, Deus.
8. Fílii autem hóminum * in tégmine alárum tuárum sperábunt.
9. Inebriabúntur ab ubertáte domus tuæ: * et torrénte voluptátis tuæ potábis eos.
10. Quóniam apud te est fons vitæ: * et in lúmine tuo vidébimus lumen.
11. Præténde misericórdiam tuam sciéntibus te, * et justítiam tuam his, qui recto sunt corde.
12. Non véniat mihi pes supérbiæ: * et manus peccatóris non móveat me.
13. Ibi cecidérunt qui operántur iniquitátem: * expúlsi sunt, nec potuérunt stare.

Antiphona

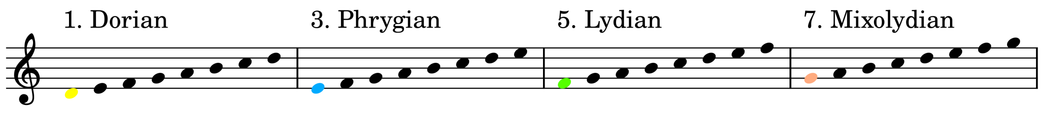
Con-trí-tum est cor me-um in mé-di-o me - i, contremu-é-runt ómni - a ossa me-a.

Figure 1.2: Example of the antiphon, its psalm starting with the Ps. 35, and again repeated the antiphon from the beginning. The differentia is a melodic unit at the end of the psalm. Psalms are usually not part of tonaries. They are encoded at the end of antiphons as differentiae. (Hiley [2009])

block: an octave was achieved as two such tetrachords put adjacent to each other by a tone. Thus, the difference between modes lies in the permutation of the scale, or rather in the placement of the two semitones of the “white key” scale in relation to the final note. The eight modes are grouped into four pairs, each with a different final tone. One mode in each pair is referred to as *authentic*, while the other is called *plagal*, distinguished at least by their respective range of tones. The exact range is not precisely defined. There is only provided the approximation based on chant practice containing several exceptions (Hiley [2009, p. 168]). The names of the modes originated from Byzantine terminology, the *oktoechos*, but by the end of the ninth century, other terms inspired by Greek music theory were also observed, most notably in the *Alia musica*[2] treatise. Table 1.1 provides a summary of the fundamental properties of all eight modes. It is important to note that the finalis and range data in the table are based on chant practices, representing approximations rather than strict rules.

	tonaries, Aurelian, etc.	Alia musica	finalis	conventional range
1	protus authentus	Dorian	D	C-c
2	protus plagalis	Hypodorian	D	A-a
3	deuterus authentus	Phrygian	E	D-d
4	deuterus plagalis	Hypophrygian	E	C-c
5	tritus authentus	Lydian	F	F-f
6	tritus plagalis	Hypolydian	F	C-d
7	tetradus authentus	Mixolydian	G	F-f
8	tetradus plagalis	Hypomixolydian	G	C-d

Table 1.1: Summarization of modes and their tonary names, Alia music names, final tones, and ranges by Hiley [2009]



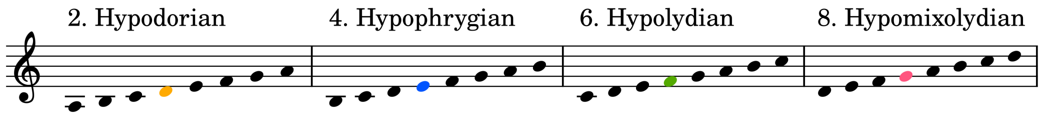


Figure 1.3: Mode scales with the colored finales. Modes are more complex than scale definitions, as we will show later in this work.

The modal system plays a crucial role in understanding the significance and functions of tones in melodies. Each chant was composed in a specific mode, facilitating better categorization of chants and aiding in their more accessible learning

[2]`https://www.oxfordreference.com/display/10.1093/oi/authority.20110803095402443`

and memorization. The organization of individual chants into modes (and their subcategories) is known most prominently from a category of chant books called *tonaries*. Furthermore, in many churches, chants in the repertoire were grouped according to the mode associated with a particular day. For example, it was quite common to sing chants of the first mode on the first Sunday, followed by chants of the second mode on the subsequent Sunday, and so forth (Hiley [2009, p. 169]).

One possible reason for developing modes could be establishing connections between antiphons and their corresponding psalms, or responsories and associated verses. It was common practice for each psalm to be sung mainly in a specific tone assigned to its mode. For each psalm tone (which also means for each mode), a limited set of possible psalm cadences existed. Therefore antiphons were secondarily grouped in tonaries by these cadences of related psalms. The grouped antiphons shared similar melodies, thereby forming distinct melodic families. One could anticipate the psalm tone and its cadence by learning these melodic groups. Responsories were slightly more complex, but their verses sung after the lessons in the Night Office were similarly assigned a specific tone based on the mode. This is a practical perspective of modality as a principle of repertoire organization, potentially useful for memory, which had not much to do with the theoretical perspective until at least the 11th century. The idea that established melodic units are related to modes, or that each mode could also be expressed as a "glossary" of established melodic units, comes from this empirical perspective.

Despite the attempt at precisely describing the modal theory, it is essential to note that many chants did not strictly adhere to these rules. Some theories claim that these are just mistakes of the singers, who couldn't remember all the notes the same way and modified some notes a bit. In some cases, attempts were made to correct the melodies. Additionally, when new chants were composed for a particular occasion just established, they often aligned more closely with described modal rules.

1.3 Chant Complexity

As already indicated, the Gregorian chant was built on a complex system that included more than just modality. The significant characteristic of chant melodies also lies in their texts. Chant texts draw inspiration from biblical texts and Church Fathers' commentaries. While the chants primarily incorporate paraphrased or quoted lessons, their meaning deviates slightly from the original lessons. Chant texts often include additional biblical passages or entirely new individual thoughts. However, the purpose of chant texts is more about continuing the narrative of the Bible rather than delivering sermons. It is important to note that each chant is not an isolated component but is interconnected with other chant texts. For example, responsories often respond to preceding parts of the service, and so on.

The text of the chants is connected to the melodies in some way. The end of each textual phrase often corresponds to one of the well-known cadences. As mentioned earlier, these cadences are closely tied to the chant modes. However, composers faced the challenge of aligning melodies and texts throughout the entire chant. In simpler pieces like antiphons or hymns, each text syllable was matched with a single tone. More complex chants such as responsories, graduals, tracts,

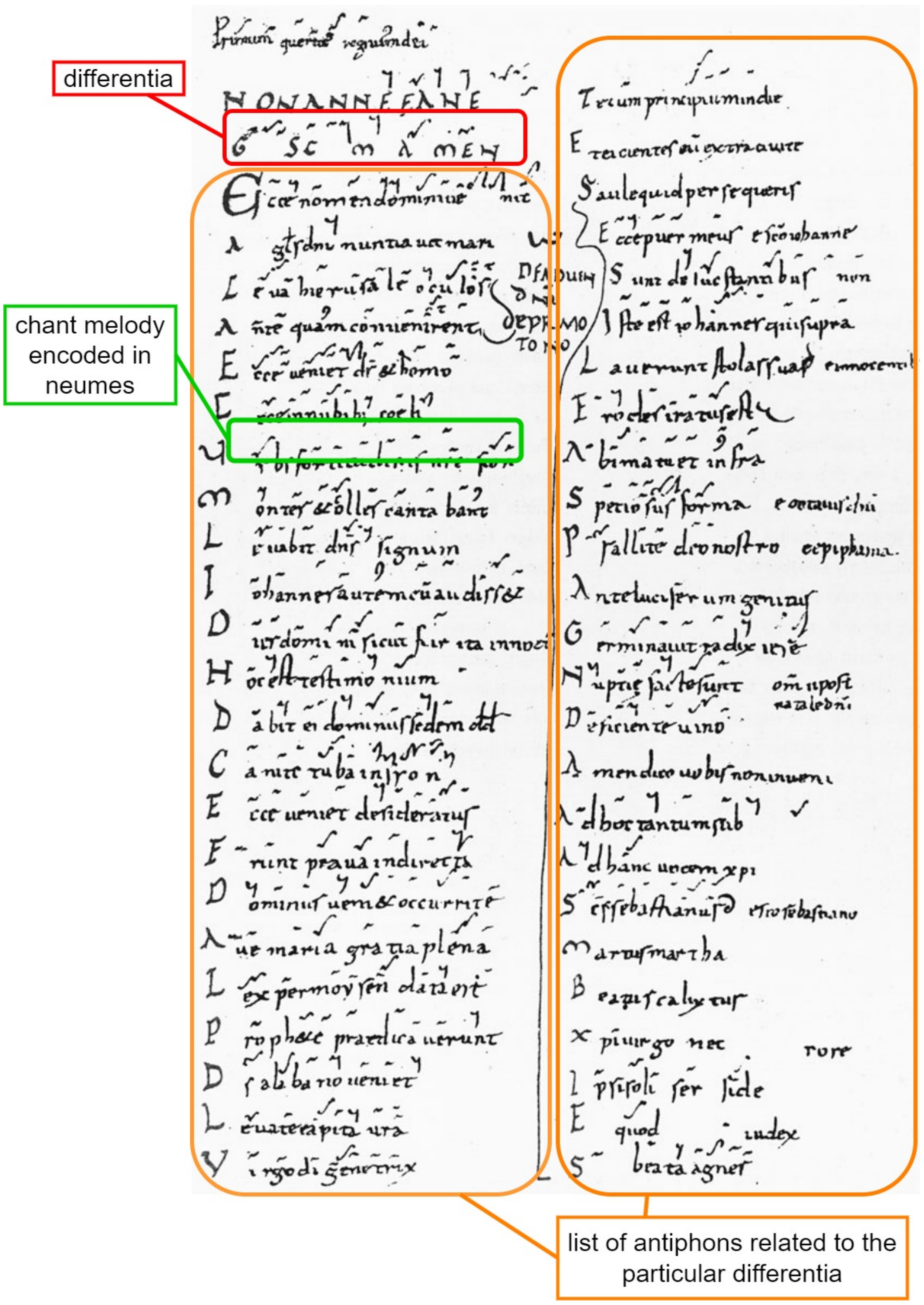

differentia
chant melody encoded in neumes
list of antiphons related to the particular differentia

Figure 1.4: The beginning of the tonary, the differentia example and its list of antiphons (Tonary of Regino of Prüm).

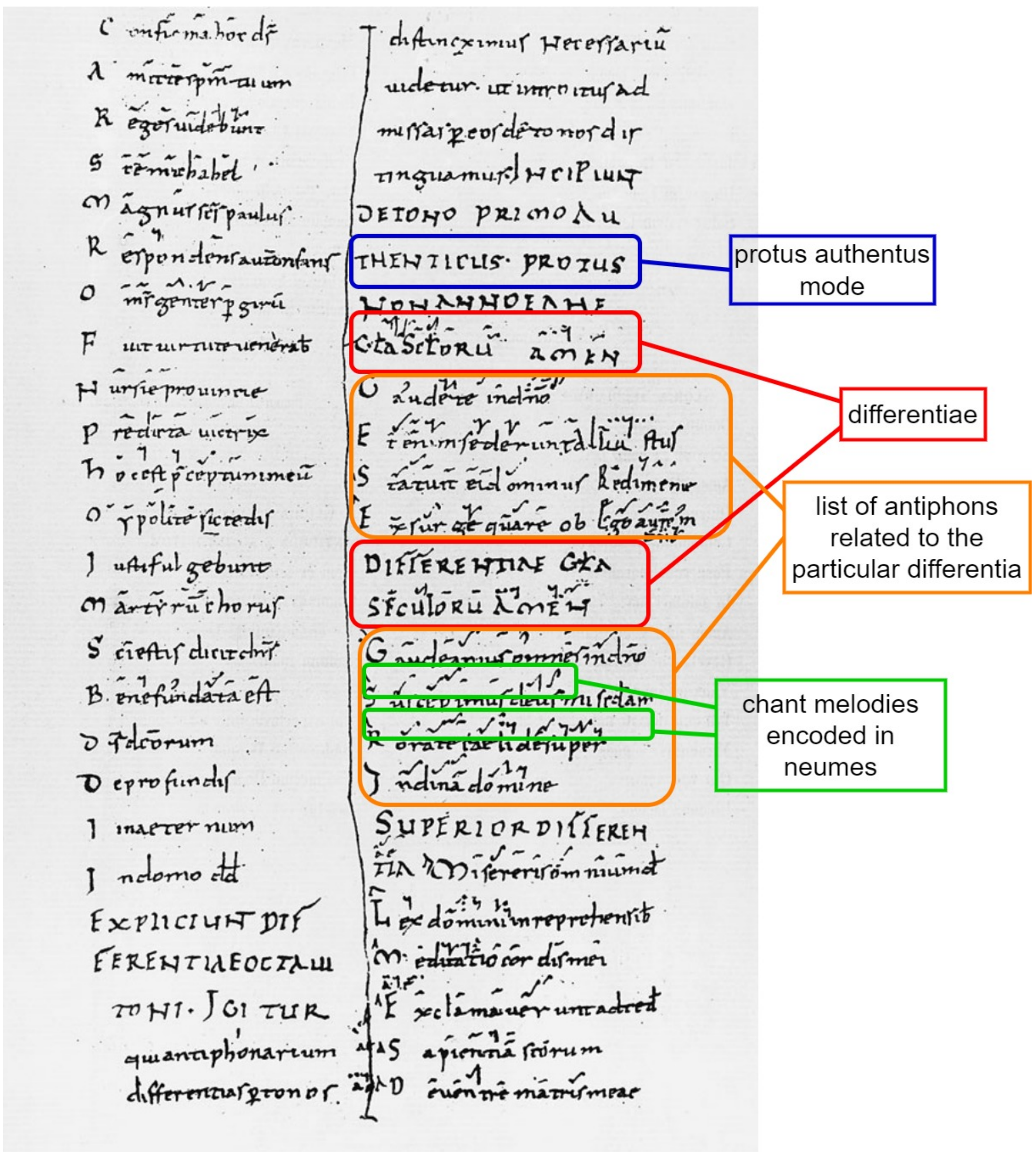

protus authentus mode
differentiae
list of antiphons related to the particular differentia
chant melodies encoded in neumes

Figure 1.5: The beginning of one of Introits, an example of two differentiae and their lists of antiphons and their melodies that belong to the first mode, protus authentus (Tonary of Regino of Prüm).

or offertories featured *melismas*, melodic passages spanning multiple tones sung within a single syllable of the text.

Another notable feature of melodies was the use of refrains or repetitions. Conversely, there were also melodies of a more straightforward character. Reference books categorize chants based on their level of simplicity. That could suggest that more complex melodies may have evolved from simpler ones over centuries. However, it cannot be definitively proven. It is possible that complex and straightforward pieces were developed and employed approximately simultaneously.

The structure of the antiphons in Office hours is relatively simple. As previously mentioned, each antiphon is repeated twice, with a psalm sung in between. The antiphons themselves are generally short, typically consisting of around four phrases. On regular weekdays, many antiphons are composed even of just two phrases. Specific melodic phrases are more commonly used than others. They only had to align with the structure and content of the accompanying text. However, some of these more familiar melodies may sometimes be slightly modified.

Antiphons and responsories share many similarities but also have some differences, particularly in melodic richness. A responsory consists of two parts: the response and the verse. The response initiates the chant, followed by the verse, and then the second part of the response is repeated at the end. This structure is reminiscent of antiphons, which frame psalms. Both responsory verses and psalms employ characteristic tones for recitation and feature final cadences. Furthermore, responsory verses are divided into two halves, each with its own distinct structure. For example, each half may have different reciting tones. However, responsory verses still tend to have more complicated melodies compared to psalms. Responsory responses are longer than antiphons. They typically consist of six phrases that are often organized in two-phrase periods. Similarly, there are certain melodic phrases in responsories that are commonly used, although they may be modified to accommodate variations in the number of text syllables. Nevertheless, the final cadences in responsories are generally preserved without significant alterations.

1.4 Oral Tradition

From its inception, Gregorian chant relied on oral tradition. The earliest chant books emerged in the late eighth and early ninth centuries but did not include specific melodic notation. Instead, they featured *neumes*, symbols indicating the number of tones per syllable, the melodic direction, and occasionally hints of rhythm or stress. They encoded just the shape of the melody rather than the exact pitches, acting thus likely as memory aids. Pitches started being notated exactly with the advent of staff notation in the mid-11th century. However, these books still primarily served as teaching aids for chant cantors. Regardless of the availability of such books, singers throughout the middle ages have had to learn chants by memory. While many chants have melodies similar to each other, especially within the "melodic families" found in individual modes, the melodies are still unique and do require memorization individually.

Historical records of weekly tables were created to outline the responsibilities of all singers. These tables divided the chants of the week among the performers, making it easier to memorize or perform difficult pieces. However, the number of chants that singers had to learn was still enormous. Additionally, teachers themselves had to remember melodies before the advent of written notations. It is reasonable to assume that a sophisticated system existed for learning and memorizing these melodies. One hypothesis is that the system combines familiar melodic units according to specific rules, which has been to some extent observed (Ferretti [1934], Helsen [2008]), but the extent to which this principle applies is unknown.

The oral tradition seems to have worked well, but not without flaws. In the ninth century, books of chants for the Proper of the Mass throughout the annual cycle were developed. Surprisingly, these books were nearly identical. However, there are differences, and another major unknown in chant scholarship is whether this diversity can be attributed to mistakes in the cantor's memory or personal preferences, and thus essentially chance, or whether it represents some "melodic dialects" according to the region or ecclesiastical context.

1.5 Digital Chant Representations

Many melodies have already been digitized, especially for the Cantus Database (Lacoste [2022]). New chant representations had to be able to encode elements like clefs, text details, barlines, and individual pitches. Two format types have been proposed for this purpose. The first is the *gabc* notation, employed mainly in the GregoBase database by Berten [2013]. However, in this work, the *Volpiano* font will be predominantly utilized.

1.5.1 Volpiano Font

The font Volpiano is used to represent Gregorian chants as strings, consisting mainly of alphanumeric characters and hyphens. The string starts with a clef sign followed by three hyphens. For instance, the treble key is denoted as `1---`. After the clef sign, characters are used to symbolize specific pitches. Each pitch can be represented by one of two signs: the first represents the default pitch, while the second represents *liquescent* neumes. Liquescent neumes correspond to specific letter combinations in the text, such as double consonants or diphthongs. Figure 1.6 illustrates the pitch representations and their liquescent variations.

Additionally, the Volpiano font includes information about the text. A single hyphen between pitches signifies two separate neumes sung to the same syllable, while two hyphens indicate the second pitch sung with another syllable. Lastly, three hyphens indicate the second pitch sung with an entirely new word.

Gregorian chant scores also include several types of barlines. A single barline is symbolized by `---3`, while a double barline is represented by `---4`. The double barline is used, for instance, to denote differentiae in antiphons. Other barlines, such as middle barlines or commas, are represented using numbers `6` or `7`. The Volpiano font is primarily used for encoding chants in the Cantus database by Lacoste [2022]. Suppose we convert the gabc notation, primarily used for encoding chants in the GregoBase database by Berten [2013], into the Volpiano font. In that case, these bar lines generally have a different meaning. We then refer to them as pause types.

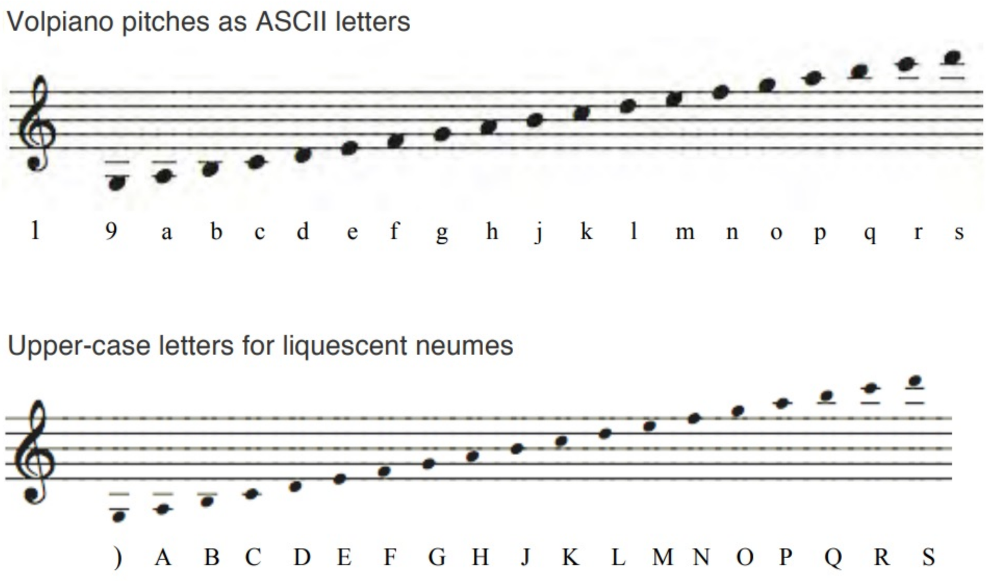

Volpiano pitches as ASCII letters
1 9 a b c d e f g h j k l m n o p q r s
Upper-case letters for liquescent neumes
) A B C D E F G H J K L M N O P Q R S

Figure 1.6: Pitch representations in Volpiano font, taken from Swanson et al. [2016]

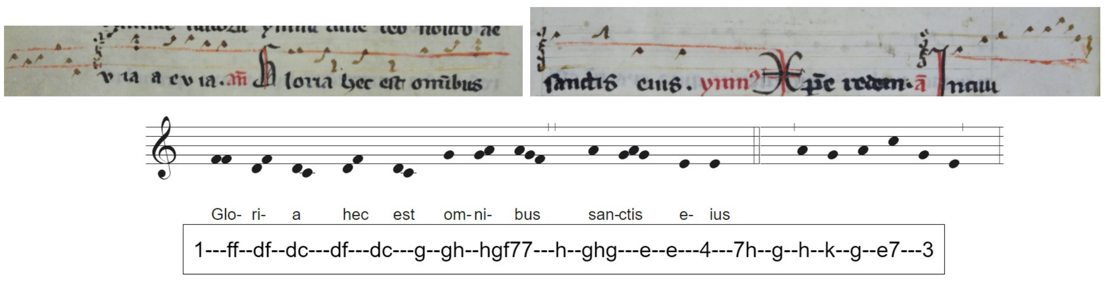

Glo- ri- a hec est om- ni- bus san-ctis e- ius
1---ff--df--dc---df---dc---g--gh--hgf77---h--ghg---e--e---4---7h--g--h--k--g--e7---3

Figure 1.7: An example of the chant “Gloria haec est omnibus sanctis” taken from Lacoste [2022]. The first two images display the original notation, while below them, you can see the same chant represented in today’s notation using the Volpiano font.

2. Statistical Background

In this chapter, we will provide a brief overview of the mathematical methods used in this thesis. These are unsupervised techniques using Bayesian statistics, as well as those employing neural networks. Additionally, we will briefly describe the two machine learning classifiers that we use for mode classification.

2.1 Unsupervised Techniques

Unsupervised methods are used in situations where the task is not to predict some target values but to provide a probabilistic model of a phenomenon represented by some (presumably large) collection of training data. Nowadays, many tasks have been successfully solved using models that learn from large amounts of unlabeled data, commonly referred to as training data, in a particular domain of interest. These models aim to discover and learn the patterns within the observed data, enabling them to apply this knowledge to unseen samples.

There are several distinct types of unsupervised tasks, with clustering and dimensionality reduction being the most common. The unsupervised segmentation task is a problem based on a probabilistic model about which we have some assumptions (memory efficiency), and based on this, we define the different distributions in the model and their connections, and training is the process of finding the parameters of this model so that the probability of the data is maximized according to the defined model. There are two approaches to consider. The first approach involves utilizing Bayesian statistics, while the second involves using neural networks. Neural networks have proven very effective but require large amounts of learning data. While they introduce more assumptions and may not be as universal and powerful, Bayesian methods can yield satisfactory results with less data. Additionally, working explicitly with probabilities of segmentations is much more accessible within the Bayesian models.

2.1.1 Bayesian Statistics

Bayesian statistics is one of the statistical approaches motivated by providing the framework that enables belief updates based on available evidence while quantifying uncertainty. This framework combines a *prior* distribution with a *likelihood* function to obtain the *posterior* distribution. The prior is a distribution specified before observing any data, and it is an initial belief about model parameters. On the other hand, the likelihood function gives the probability of our data given specific parameters. The resulting posterior represents the probability of the model parameters given the data. The goal is usually to find parameter values with the highest probability according to the posterior distribution. In other words, it contains information about how to update the parameters. The formulation of this framework relies on the utilization of Bayes' theorem.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.1}$$

In the context of Bayesian statistics, A stands for our data, and B could be replaced by model parameters θ. Then we can rewrite the equation as:

$$P(\theta|data) = \frac{P(data|\theta) \cdot P(\theta)}{P(data)} \tag{2.2}$$

In this context, the term $P(data)$ refers to the marginal likelihood independent of the specific parameters. This implies that the value of the marginal likelihood remains constant for every set of parameters. Therefore, we do not need to take into account its value when updating the model parameters, so we have:

$$P(\theta|data) = \frac{P(data|\theta) \cdot P(\theta)}{P(data)} \propto P(data|\theta) \cdot P(\theta) \tag{2.3}$$

which gives us:

$$posterior \propto likelihood \cdot prior. \tag{2.4}$$

MacKay [2003] provides more detailed insight into Bayesian inference. However, the choice of the prior distribution is tied closely to the likelihood function. Very often, the model manipulation would be more straightforward if the posterior kept the same form as the prior distribution. We call such prior distribution a *conjugate prior* to the likelihood function. Table 2.1 shows some examples of conjugate priors.

Likelihood	Conjugate prior	Posterior
Bernoulli/Binomial	Beta distribution	Beta distribution
Poisson/Exponential	Gamma distribution	Gamma distribution
Multinomial	Dirichlet distribution	Dirichlet distribution

Table 2.1: Examples of conjugate priors based on Fink [1997]

Bayesian Nonparametric Models

Orbanz and Teh [2010] classify models in Bayesian statistics into two categories. The first category consists of parametric models, which rely on a fixed number of parameters. Examples of parametric models include, for instance, hidden Markov model (HMM) and Linear Regression. These approaches are suitable when we have prior knowledge about model structure: about the number of hidden states in the case of HMM, the number of latent variables in latent variable models, or when we need to specify the number of clusters for clustering problems. These models operate in an infinite-dimensional parameter space, providing greater flexibility. These models are more relevant for our particular task because we want to assign probabilities to segmentations of chant melodies while not knowing even just the size of our vocabulary of melodic units. Their size can thus expand based on the size of our data. Examples of such models include the infinite HMM, Dirichlet process, Gaussian process, Beta process, or Pitman-Yor process.

The training process of such a model is not a straightforward task because the posterior distributions are often intractable. However, there are already several algorithms searching for the best posterior distribution. This work will focus on the Gibbs sampling algorithm, one of the Markov chain Monte Carlo procedures

(MacKay [2003]). The Gibbs sampling algorithm begins by initializing the latent variables, typically with random or, in some cases, specific initial values. In the next step, each variable is sampled anew based on the values of the remaining variables. That means that when sampling a variable's new value, the algorithm ignores its old value. The sampling step is iterated until the final distribution is close to optimal. The order in which variables are sampled is randomly shuffled in each iteration. After the iterations, the resulting sampled variables are used to estimate the posterior distribution. Gibbs sampling is thus a natural fit for sequential models with limited context dependencies, such as n-gram models, because one can simply iterate over a permutation of the tokens in the sequence.

After training the model, obtaining predictions or answers for our specific task becomes crucial. Because we work with sequences (of chant segments), the Viterbi algorithm (Forney [1973]) holds significant importance as a decoding algorithm. It uses dynamic programming to determine the most likely path in a HMM based on the provided data. The Viterbi algorithm considers all potential paths that could be the most probable and finally identifies the best path when it reaches the end of the sequence. It then retraces its steps from the end to the beginning, utilizing precomputed paths information to extract the best sequence of states, as shown in Figure 2.1. Once the model can be interpreted as HMM with states and probabilities, we can apply the Viterbi algorithm to decode the maximal likelihood prediction. It is not straightforward to interpret the segmentation as HMM, but we will deal with the problem in Section 6.1.

```
Algorithm Viterbi Algorithm($\{o_1, ..., o_T\}, \{q_0, q_1, ..., q_N, q_F\}$)
Create a table viterbi[T][N+2]                                   ▷ Create DP table
for each state $q$ from 1 to N do                                ▷ Initialization
    viterbi[1][q] $= p(q|q_0) \times p(o_1|q)$
    backpointer[1][q] $= 0$
end for
for $t = 2$ to $T$ do                                            ▷ DP recursion step
    for each state $q$ from 1 to N do
        viterbi[t][q] $= p(o_t|q) \cdot \max_{q'=1}^{N}$ viterbi$[t-1][q'] \cdot p(q|q')$
        backpointer[t][q] $= \arg\max_{q'=1}^{N}$ viterbi$[t-1][q'] \cdot p(q|q')$
    end for
end for
viterbi[T][$q_F$] $= \max_{q=1}^{N}$ viterbi$[T][q] \cdot p(q_F|q)$          ▷ Termination
backpointer[T][$q_F$] $= \arg\max_{q=1}^{N}$ viterbi$[T][q] \cdot p(q_F|q)$
return the best path by following the backtrace of the backpointers through time
```

Figure 2.1: Pseudocode of the Viterbi algorithm (Zeineldeen [2018]).

Dirichlet Process

Let's start with the Chinese restaurant motivation of the Dirichlet process. The structure of the language model based on the Chinese restaurant process consists of infinite restaurant tables that new customers could choose from. The first customer chooses any table that is currently empty, so the first customer opens a new table. Other coming customers either join an opened table shared with a particular table's group of customers or open a new table where they will sit alone for now. Each table serves a particular dish that customers are interested in. Each customer is looking for one specific dish, but there could be more tables serving the dish. Furthermore, the customer can open a new table serving the dish. This described structure is analogous to the Dirichlet process, where we have an infinite dimensional random distribution (number of tables and their customers). When we find in training data a new observation (new customer) of the specific dish, the customer's table we expand is chosen randomly proportional to:

$$\begin{cases} \frac{c_{x,k}}{\alpha+c} & \text{select table } k \text{ serving dish } x \\ \frac{\alpha}{\alpha+c} \cdot p(x|H) & \text{open a new table,} \end{cases} \tag{2.5}$$

where x is the customer's dish, c is the count of all customers in a restaurant, $c_{x,k}$ is the number of customers sitting at the table with index k serving the dish x, $p(x|H)$ is the probability of the dish x regarding the base distribution H, and α is the concentration hyperparameter. The probability of the dish x regarding the Dirichlet process model Θ is:

$$p(x|\Theta, \alpha) = \frac{c_x}{\alpha + c} + \frac{\alpha}{\alpha + c} \cdot p(x|H), \tag{2.6}$$

where c_x stands for the count of all customers of the dish x.

Pitman-Yor Process

The Pitman-Yor process is one of the most robust Bayesian nonparametric models. As Teh and Jordan [2010] describe, it is a generalization of the Dirichlet process with two parameters, discount d, and concentration α, where $0 \leq d < 1$ and $\alpha > -d$. Formally, for the Pitman-Yor process G, we have:

$$G \sim PY(d, \alpha, H) \tag{2.7}$$

where H stands for the base measure. Regarding the Chinese restaurant process analogy, the Pitman-Yor process provides the more complex system of choosing a table for the new customer using the discount factor d, so the table is randomly chosen proportional to:

$$\begin{cases} \frac{c_{x,k}-d}{\alpha+c} & \text{select table } k \text{ serving dish } x \\ \frac{\alpha+d\cdot t}{\alpha+c} \cdot p(x|H) & \text{open a new table,} \end{cases} \tag{2.8}$$

where t is the number of all tables with at least one customer. As we can observe, the discount factor of the Pitman-Yor process forces the model to have more tables for the particular dish, one with many customers and many sparsely occupied. Then, the probability of the customer corresponding to the dish x based on the Pitman-Yor process model Θ is computed as:

$$p(x|\Theta, d, \alpha, H) = \frac{c_x - d \cdot t_x}{\alpha + c} + \frac{\alpha + d \cdot t}{\alpha + c} \cdot p(x|H), \tag{2.9}$$

where t_x gives the number of tables serving the dish x.

Hierarchical Pitman-Yor Process

In sequential modeling, order is often essential. The Dirichlet process, as well as the Pitman-Yor process, are bag-of-words models that do not capture the sequential context. Therefore, it is necessary to include the context information, which is done by the hierarchical version of these models. In the context of the Dirichlet process, suppose we have two levels of tables. The first is based on the unigram distribution, while the second is based on the bigram distribution, as shown in Figure 2.2. The unigram tables are used as the base measure for the bigram distribution. Then, the probability of the dish x with the context h, concentration α, shorter context $h^{'}$, and the seating assignments Θ is computed as:

$$p(x|\Theta, \alpha, h) = \frac{c_{h,x}}{\alpha + c_{h,\cdot}} + \frac{\alpha}{\alpha + c_{h,\cdot}} \cdot p(x|\Theta, \alpha, h^{'}), \tag{2.10}$$

where $c_{h,x}$ stands for the number of customers sitting at the table of dish x with the context h, and $c_{h,\cdot}$ is the sum of customers over all tables with the context of h.

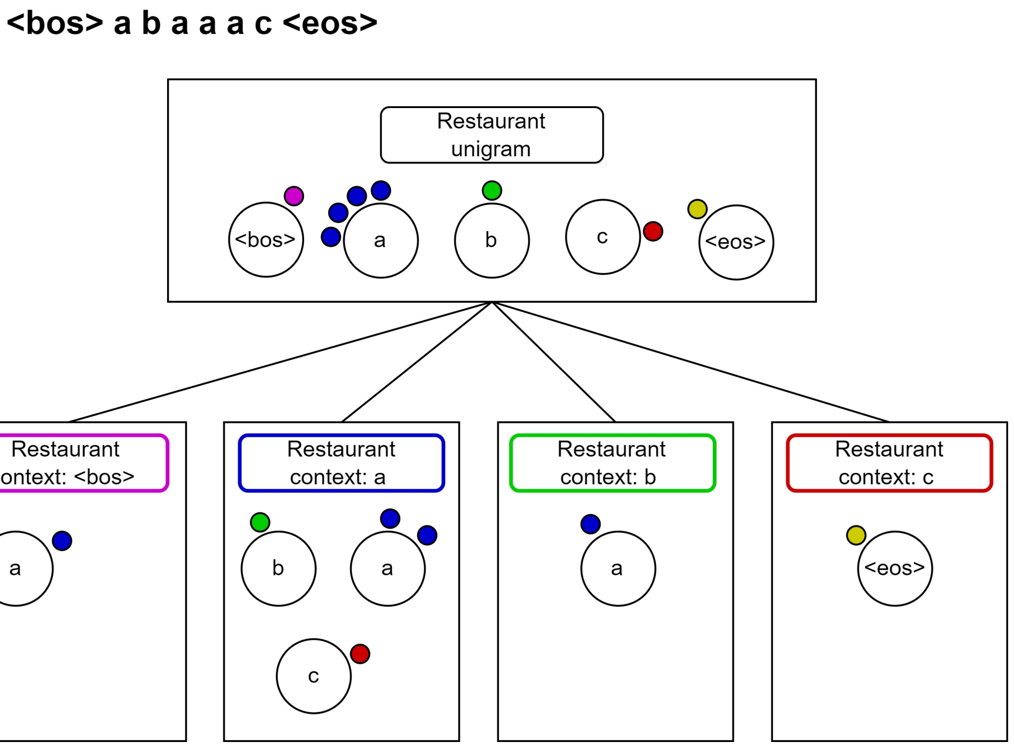


Figure 2.2: Seating assignment of the sentence "*<bos> a b a a a c <eos>*" in the Chinese restaurant analogy of the Dirichlet process.

To obtain the hierarchical Pitman-Yor process model, we have to again include the discount factor. Teh and Jordan [2010] define the hierarchical version of the Pitman-Yor process as:

$$\begin{aligned} G_0|\eta, \gamma, H &\sim PY(\eta, \gamma, H) \\ G_j|d, \alpha, G_0 &\sim PY(d, \alpha, G_0) \quad \text{for } j \in \mathcal{J}. \end{aligned} \tag{2.11}$$

Here, G_0 represents the shared base measure across all data group models, $\mathcal{J}$ denotes the set of indices for these models, and η and γ are parameters associated with the base measure model.

Although Teh and Jordan [2010] explain two types of representation of the Pitman-Yor process, where the first one is known as the stick-breaking construction, the one based on the Chinese restaurant franchise is the more intuitive one, even in the context of hierarchical Pitman-Yor process. Let's consider the random variable θ_{ji} distributed according to G_j. With the Chinese restaurant analogy, the variable represents the i-th customer in the j-th restaurant. Another random variable θ^*_{jt} corresponds to the j-th restaurant and its t-th table, which are distributed according to the base model G_0. Another set of random variables, θ^{**}_k, follows the base measure H and can be thought of as dishes in the restaurant. Then, the n_{jtk} denotes the number of customers with dish k at table t, which is in the restaurant j. The m_{jk} is the number of tables that are placed in the restaurant j with the dish k. When indices are replaced by $\cdot$, it indicates marginalization over the missing index. The vocabulary size, or the number of unique dishes, is denoted as K. Then the conditional distributions of hierarchical Pitman-Yor random variables are as follows:

$$\begin{aligned} \theta_{ji}|\theta_{j1},...,\theta_{j,i-1},\alpha,d,G_0 &\sim \sum_{t=1}^{m_{j\cdot}} \frac{n_{jt\cdot}-d}{\alpha+n_{j\cdot\cdot}}\cdot\delta_{\theta^*_{jt}} + \frac{\alpha+m_{j\cdot}\cdot d}{\alpha+n_{j\cdot\cdot}}\cdot G_0 \\ \theta^*_{jt}|\theta^*_{11},...,\theta^*_{1m_{1\cdot}},...,\theta^*_{j,t-1},\gamma,\eta,H &\sim \sum_{k=1}^{K} \frac{m_{\cdot k}-\eta}{\gamma+m_{\cdot\cdot}}\cdot\delta_{\theta^{**}_k} + \frac{\gamma+K\cdot\eta}{\gamma+m_{\cdot\cdot}}\cdot H. \end{aligned} \tag{2.12}$$

2.1.2 BERT

Bidirectional Encoder Representations from Transformers, the so-called BERT, was introduced by Devlin et al. [2019] as a powerful model for natural language processing tasks. It utilizes a transformer architecture, as depicted in Figure 2.3, to capture bidirectional context from neighboring inputs. BERT's strength lies in its ability to extract representations that very well reflect the semantics of individual tokens and entire phrases. The training of BERT involves two main phases. The first one is pre-training, and the second one is fine-tuning. During pre-training, BERT learns from vast amounts of unlabeled data, which is often easier to collect. A crucial aspect of pre-training is masked language modeling, where random input tokens are masked, and the model is trained to predict them backward accurately. This process helps BERT learn contextual representations and understand the relationships between tokens, which could be, for instance, characters or even words. The pre-trained BERT model is further trained on specific tasks in the fine-tuning phase by adding the additional layer on top of the pre-trained model. In this phase, we use mostly labeled data, but we don't need as much of it as in the masked language modeling part.

2.2 Classifiers

A classification task is a common type of problem in machine learning. The goal is to accurately predict a class for each data sample. Numerous models have been developed to tackle such tasks, but we will focus on two of them in this work. The first model is Naive Bayes, a simple and fast statistical approach that performs well when dealing with large independent data (Jadhav and Channe [2016]). On the other hand, we mention Support Vector Machine, which works well with complex data that may lack structure or may miss some information (Akkaya and Çolakoğlu [2019]).

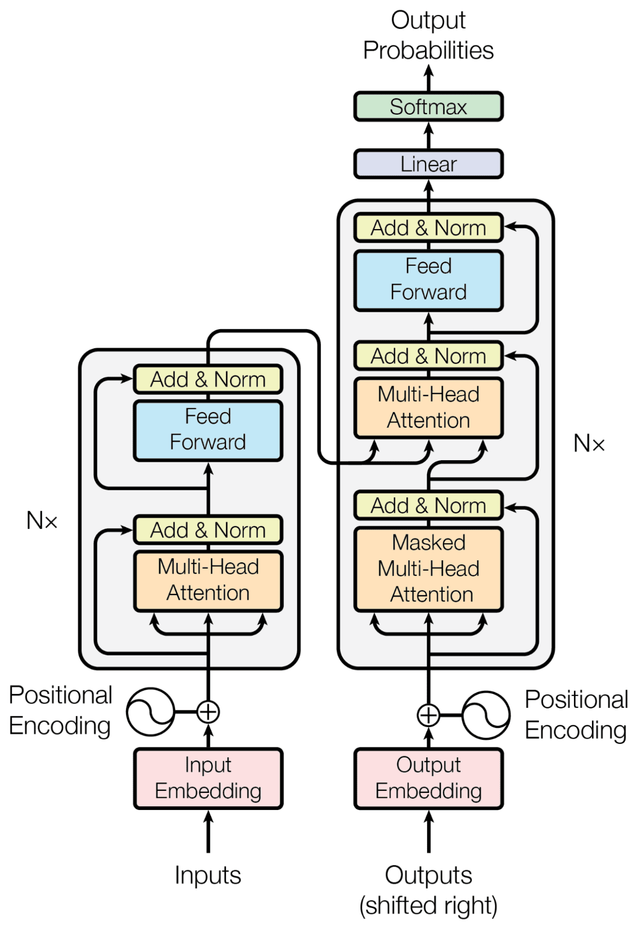


Figure 2.3: Transformer architecture (Vaswani et al. [2017]).

Some classification tasks involve working with text data to make predictions. One way to represent text as a set of features is through Term Frequency-Inverse Document Frequency (TF-IDF) (Ramos [2003]). Each document is represented as a vector, where each value corresponds to a specific term. The value is then computed as:

$$term\_frequency \cdot inverse\_document\_frequency. \tag{2.13}$$

The term frequency measures the occurrence of the term in a particular document, while the inverse document frequency represents the importance of the term across all documents, measured as the inverse frequency of documents containing the specific term. In the context of Gregorian chants, the terms could refer to melodic units, and the documents could be individual chant melodies.

2.2.1 Naive Bayes

Bishop [2006] describe Naive Bayes as a classification method based on Bayes' theorem.

$$P(C|D) = \frac{P(D|C) \cdot P(C)}{P(D)} \tag{2.14}$$

Let's consider C as a specific class and D as a data vector. In Naive Bayes, $P(C)$ represents the prior distribution of class C, while $P(D|C)$ denotes the probability of observing the data vector D given that class C, the so-called likelihood. The goal is to maximize the posterior distribution $P(C|D)$ to determine the predicted class C. The Naive Bayes method assumes that each feature contributes independently to the final probability of the potential class, so the feature probabilities could be multiplied together. The model could lead to incorrect results if the independence assumption is not met. Although Naive Bayes is a simple model that cannot handle complex features, it is quite efficient, especially when dealing with high-dimensional inputs.

2.2.2 Support Vector Machine

The Support Vector Machine method, sometimes called Support Vector Classifier (SVC), is also based on statistical theories. It involves projecting the training data into an n-dimensional space, where n is a number of features. The aim is to divide this space into distinct classification classes by so-called support vectors. At the same time, the model strives to maximize the distance between classification classes and these support vectors. When dealing with linearly separable classes, the theory behind the separation is straightforward. However, in cases where the classes are not linearly separable, the space is transformed into a higher-dimensional space using so-called kernel functions. The type of kernel function is one of the SVC hyperparameters. The transformation is the element that allows the SVC model to handle complex data effectively. (For more on SVMs, see chapters 6 and 7 of Bishop [2006].)

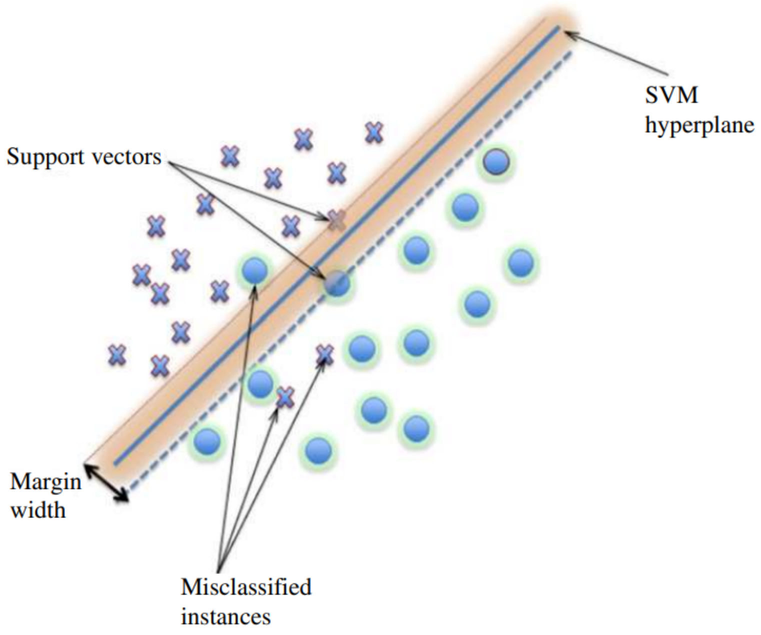


Figure 2.4: Visualization of SVC with its terminology (Mountrakis et al. [2011]).

3. Related Works

Various theories have been put forth to tackle the issue of segmenting the Gregorian melody by analyzing chant's compositional processes. It is still unclear whether it makes sense to segment melodies and, if so, whether it makes sense to segment them on the compositional level. Treitler [1975] compares different approaches and highlights the significant ambiguity surrounding the concrete rules of compounding specific melodic units into a final melody. They also mentioned that the inverse procedure could explain repetitive formulas and melodic units, the repeated performance-composition process. Performers memorized only beginnings and some fixed goals, and the rest was naturally reconstructed by them, which invented a new chant melody. However, no one has yet tried to solve it statistically. Also, other tasks are dealt with using unsupervised segmentation. For instance, we can get inspiration from general melody segmentation, topic segmentation, or Chinese word segmentation topics. On the other hand, researchers have been dealing with Gregorian chants in recent years, and we discuss and build on their work as well.

3.1 Gregorian Chants

First, the general theory and historical context of Gregorian chants are most recently and concisely described in Hiley [2009]. The work includes information about chant types or mode structures and the chant evolution over centuries. In connection with the fact that related data began to be collected and the computer science community focused on Gregorian chants began to emerge, the *Chant 21* Python library was created and described by Cornelissen et al. [2020a], which should help the community to work with chants and their formats.

Also, some research works and experiments have already been done. A significant contribution was the work by Cornelissen et al. [2020b], which tried to classify modes using the chant's sheet music for both antiphons and responsories. Melody segmentation is one of their approaches where they used segments for creating features for the SVC model following the TF-IDF vectorizer. The segmentation approach worked best for them. They considered several possible naive segmentations - powerful ones seem to be 4-grams, 5-grams, 6-grams, segmentation by words, and segmentation by syllables. These approaches were even better than considering final tone and chant ranges, one of the significant mode characteristics from the music-theoretical perspective, thus providing quantitative evidence that the empirical perspective may explain the phenomenon of chant modality more comprehensively. For the evaluation, they used data from Cantus provided by Lacoste [2022], a database of cataloged and, in a portion of cases, also transcribed Gregorian chants. It is important to mention that the authors also included differentiae (final melodies of the following psalm) in antiphons for the mode classification task. That means that their results for antiphons are not correctly evaluated since they also used additional information about the further context, which is related to Introit structure and its modes. Therefore it is not entirely clear whether their proposed antiphon segmentations are really that strong or only segmented differentiae are that helpful.

There is also the Gregorian chant database, the GregoBase described in Berten [2013], which contains chants in gabc format. For easier manipulation of the database, there is the Chant21 Python library. As part of the chant transcriptions, there is information about the end of phrases or other pauses and part separators in scores. That could be beneficial for the task of melody segmentation.

3.2 Unsupervised Segmentation Tasks

Many other current tasks deal with unsupervised training. Most of them use Bayesian statistical methods, which have been researched and improved in the last 20 years. On the other hand, there are also several experiments based on Bidirectional Encoder Representations from Transformers (BERT) by Devlin et al. [2019].

3.2.1 Topic Segmentation

One of the most important articles on topic segmentation problems based on the Bayesian processes was published by Eisenstein and Barzilay [2008]. The work combines lexical cohesion with the Latent Dirichlet Allocation work described by Blei et al. [2003].

A few years later, there was an attempt to use BERT and A Robustly Optimized BERT Pretraining Approach (RoBERTa) systems for the topic segmentation task by Solbiati et al. [2021]. Utterances were input to those systems, and the systems tried to predict the sequence of ones and zeros, one for the topic boundary and zero for the no-boundary scenario. The segmentation and training are based on the cosine similarity of two neighboring utterance blocks. The Bayesian method by Eisenstein and Barzilay [2008] still performs better. Another article exploring BERT is published by Xing and Carenini [2021]. They researched and designed a new score function based on utterance pairs.

3.2.2 Word Segmentation

Word segmentation problem has a lot in common with melody segmentation, especially when we want to segment already transcribed tones. On the other hand, the word segmentation problem is much more explored.

A major improvement in the word segmentation problem was brought by utilizing Pitman-Yor process - the Bayesian method generalizing the Chinese restaurant process. Teh [2006a] described the hierarchical Pitman-Yor language model (HPYLM), the hierarchical Bayesian model where the structures store the previous context information of the current word or character. Mochihashi et al. [2009] used the inspiration of the previous work and trained the nested hierarchical Pitman-Yor language model (NHPYLM) model using the Blocked Gibbs Sampling algorithm with Poisson correction to make the algorithm prefer segments with the most common length. The NHPYLM was combined from word HPYLM and character HPYLM. The character one was used as a prior for the word one.

Also, there were attempts to use the BERT for this task. The article published by Li et al. [2022] uses discriminative and generative modules. The generative module trains the model to minimize the distance between two neighboring characters from the same segment and to maximize the distance between two adjacent characters from different segments. The discriminative module only works with labels B and I as beginning and inside segment positions, respectively, via the masked language modeling technique. This model is currently state-of-the-art in word segmentation tasks, but we suspect training BERT would require larger datasets than what we have available for chant research.

3.2.3 Melody Segmentation

Researchers have been working on possible melody segmentation approaches for decades. Lerdahl and Jackendoff [1983] described rules that generate melody segments. There were also other attempts to solve the task. Pearce et al. [2010] used Information Dynamics of Music (IDyOM), variable-order Markov model, to find music context and structure statistically, which had better performance than previous rule-based methods. A similar approach had Lattner et al. [2015], except they were experimenting with the Restricted Boltzmann Machine (RBM) model. There is also the supervised research of melody segmentation by Guan et al. [2018] considering the convolutional neural networks - conditional random field (CNN-CRF) model.

Several musically focused articles have experimented with the Pitman-Yor process as well. One of them is researching the melody segmentation problem published by Sawada et al. [2020]. They used the motif-level bigram HPYLM using the note-level HPYLM as a base measure. They did the same thing as Teh [2006a] and Mochihashi et al. [2009] did for words. They considered several representations, such as pitch classes, durations, or intervals, where durations give the best scores. For the final evaluation, they compared nested Pitman-Yor language model results with segmentations generated by grouping preference rules and grouping structure described by Lerdahl and Jackendoff [1983] using the F-measure score. The F-measure was highest for the duration sequence compared with the grouping structure method, which was 34.2. For us, the result of 29.4 of the pitch representation compared with the grouping structure method is more relevant since we do not have information about Gregorian chant note durations.

Another possible scoring function could be the usage of the technique based on probabilities of misses, false alarms, and disagreements introduced by Beeferman et al. [1999]. Then the segmentation of our proposed model could be compared with some manual segmentation, random segmentation, and n-grams, the same way as did Melucci and Orio [2002].

4. Datasets

We need a significant amount of data to estimate the parameters of non-trivial models with a chance of success. The availability of transcriptions of manuscripts or early printed books is helpful in this regard. We primarily use the Cantus database (Lacoste [2022]) as a source of data. More data is also available via the GregoBase database (Berten [2013]). Each of them is structured differently, and both are focused on slightly different information. However, both need to be filtered first in order to use them properly.

4.1 Cantus Database

We have decided to use the Cantus for segmentation training and testing processes since, so far, the database is the largest digital resource of Gregorian chants. It contains a wealth of data about the contents of thousands of medieval manuscripts. The dataset includes several details about each chant. Melodies, genres, and modes are most crucial of them for the goal of our research. The Cantus stores melody information in Volpiano format. However, deeper information about melodies like final tone, differentiae, or lyrics could also be found there. In the database, chants are also mapped to their source manuscripts and particular pages. Moreover, the Cantus provides details regarding the day's office and the feast of the year when the chant was performed, along with the liturgical role.

We will follow up on Cornelissen et al. [2020a], who scraped and dumped this database into the so-called Cantus Corpus dataset. Only less than 10 % of the cataloged chants have fully transcribed melodies. Therefore Cornelissen et al. [2020b] conducted filtration of the dataset before they used it for the mode classification task. We use the same filtration. Initially, they excluded records without Volpiano melodies, notes, or simple modes. Then, they removed chants with incomplete lyrics or with an incipit as the full text. Melodies that began with clef other than G, or those with missing pitches, are also discarded, as well as those that contain incorrect Volpiano characters or missed word boundaries. Lastly, they removed duplicated melodies. Also, it is necessary to mention that Volpiano data contain liquescent neumes, symbols indicating certain consonants or diphthongs. They do not affect the melody at all. Therefore we are replacing them with their pitch alternatives since those liquescents would introduce artificial data sparsity for the segmentation task. In any case, it should be noted that the Cantus Corpus dataset remains unfiltered. This thesis works with version 0.2, which contains 497,071 chants of 57 genres from 640 sources. However, we work only with genres of antiphons and responsories, reduced by the described filtration.

After filtering, 13,865 antiphons remain for our experiments. We follow the methodology of Cornelissen et al. [2020b] and use a 70:30 training/test split. The average length of an antiphon chant melody is 59.51 tones. The shortest is three pitches long, while the longest is compounded from 683 notes. The distribution of modes across the chants is shown in Table 4.1.

One additional data cleaning issue is that more than 95 % of antiphons are transcribed including their differentiae. These can significantly impact the re-

1. Dorian	3. Phrygian	5. Lydian	7. Mixolydian
3,406	950	532	2,137

2. Hypodorian	4. Hypophrygian	6. Hypolydian	8. Hypomixolydian
1,005	1,372	530	3,933

Table 4.1: Number of chants from the antiphon dataset mapped to a particular mode.

sults of both mode classification and chant segmentation since the differentia is a submelody of another chant. We keep the antiphon dataset with them since we want to compare our results with Cornelissen et al. [2020b]. But in order to have correct results and a straightforward segmentation process, we generated the second antiphon dataset, where we discarded all the differentiae from the Volpiano melody before the filtration process of Cornelissen et al. [2020b]. However, this has resulted in several new duplicated pieces. Therefore the final dataset without differentiae comprises 13,551 antiphons, 9,486 for the training dataset, and 4,065 for the testing one. Then, the average antiphon melody length is decreased to 53.97. Also, the mode distribution slightly differs, as seen in Table 4.2.

1. Dorian	3. Phrygian	5. Lydian	7. Mixolydian
3,348	939	522	2,091

2. Hypodorian	4. Hypophrygian	6. Hypolydian	8. Hypomixolydian
979	1,327	513	3,832

Table 4.2: Number of chants from the antiphons-without-differentiae dataset mapped to a particular mode.

The second genre for which Cantus Corpus v0.2 provides an amount of melodies that may prove sufficient for training our models are responsories. Overall, there are 7,031 of them, split into 4,922 for the training set and 2,109 for the testing set. The average length of a responsory chant is approximately 137.52 tones. The minimum melody length is five tones, and the maximum is 364 tones. The distribution of modes can be seen in Table 4.3.

1. Dorian	3. Phrygian	5. Lydian	7. Mixolydian
1,258	679	443	1,307

2. Hypodorian	4. Hypophrygian	6. Hypolydian	8. Hypomixolydian
904	741	227	1,472

Table 4.3: Number of chants from the responsory dataset mapped to a particular mode.

In addition, for training purposes, the training set of each of these three datasets is split into two parts. The first part comprises 90 % of the data and is used for training itself, while the remaining 10 % is reserved for validation.

4.2 GregoBase Database

GregoBase is the second largest database of Gregorian scores. There are more than 18,000 chants, which is considerably less than in the Cantus database. However, it can still be beneficial for our task. The data again contain information about where the Gregorian pieces were taken from. We could also find their modes and genres there. However, the most interesting information there is the melodies, which, unlike transcriptions in Cantus, preserve markings for pauses — breath marks or barlines. It can be assumed that most melody segments have not gone through the pause marking. These melodies are also encoded differently, using gabc format instead of Volpiano.

In this work, we will use the GregoBase Corpus version 0.4 by Cornelissen et al. [2020a], a dump from the Gregobase database. The corpus contains only 9,174 chants. In order to convert melodies in gabc format to Volpiano, we used the library by Cornelissen [2020]. Then, we processed the Volpiano format into a string of Volpiano notes and pauses, without any other additional Volpiano symbols. We use the special symbol '|' to indicate a pause. We filtered the data by genre and obtained one phrase dataset of 3,802 antiphons and another of 495 responsories. As with the previous database, we replaced liquescents with their original pitch alternatives. It is important to say that the GregoBase transcriptions don't include differentiae. Then, we don't need to generate another antiphon dataset without them.

The antiphon phrase dataset is relatively more extensive than the responsory one. It contains 20,148 phrases of antiphons, where the average phrase length is 9.97 of pitches in the range of two to 34 notes. The mode distribution is not that different from the corpus dataset, as shown in Table 4.4. There are 182 of antiphons with missing mode information.

1. Dorian	3. Phrygian	5. Lydian	7. Mixolydian
933	250	147	533

2. Hypodorian	4. Hypophrygian	6. Hypolydian	8. Hypomixolydian
282	356	175	944

Table 4.4: Number of chants from the antiphon phrase dataset mapped to a particular mode.

The responsory phrase dataset contains only 495 responsories with 7948 phrases. The average phrase length here is 12.02 pitches. The shortest phrase has only one pitch. The longest one has 63 of them. We can see the mode distribution in Table 4.5. 125 responsories are not included in the table since their mode is unknown.

Even though we do not use this dataset for training our models, we still find it useful. The dataset gives us information about phrasal breaks, so we can use it to evaluate our model's ability to predict segments that do not overlay pause markers. However, we describe this evaluation function in more detail in Section 5.2.

1. Dorian	3. Phrygian	5. Lydian	7. Mixolydian
71	27	50	32

2. Hypodorian	4. Hypophrygian	6. Hypolydian	8. Hypomixolydian
62	23	65	40

Table 4.5: Number of chants from the responsory phrase dataset mapped to a particular mode.

5. Evaluation Metrics

It is not immediately clear how the quality of segmentation of a corpus of chant melodies should be measured. In this chapter, we discuss some options: scoring functions that measure properties of a chant melody segmentation, both in relation to modality and in general. Finally, we will describe a feature extraction method to extract the most significant melodic units.

5.1 Perplexity

Perplexity is one of the most significant and fundamental evaluation metrics in probabilistic sequence modeling. It is derived from the cross-entropy of the distribution estimated on the training data and applied to the test data. It measures how well the distribution estimated on the training data generalizes to the test data, serving as an indirect indicator of how accurately the estimated distribution models the given phenomenon, as opposed to merely capturing random patterns in the training data. Lower perplexity signifies a better fit of the estimated distribution to the general unknown data, and it indicates that the model is more confident and accurate in predicting the next segment of the sequence. It indicates how confident the model is in its predicted segmentation. It is computed as:

$$perplexity = e^{-\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\bar{c}_i|}\ln(p(\bar{c}_i))}. \tag{5.1}$$

Here, N is the number of chants, $\bar{c}_i$ represents the segmentation of the chant with id i, $|\bar{c}_i|$ is the number of segments in the chant with id i, and $p(\bar{c}_i)$ is the probability that the model predicts a specific segmentation. However, this metric cannot be used for models based on neural networks.

5.2 Segmentation Scores

Since we do not have specific knowledge of what the segmentation should look like, we will analyze the model and its predicted segmentation based on general characteristics of Gregorian chant, musical principles, and statistical properties.

Vocabulary Size

One important piece of information can be the size of the final vocabulary. We obtain this by counting the number of unique segments in the final predicted segmentation. Considering that Gregorian chant is an oral tradition, a smaller vocabulary size of melodic units might be preferable as it could be easier for singers to remember. We refer to this number as *vocab_size*.

Average Segment Length

The average segment length correlates with the size of the vocabulary. Due to the fewer combinations of short segments compared to longer ones, the vocabulary of short segments tends to be naturally smaller. In the case of small average segment lengths, it could support the theory of the absence of strict rules for melodic units. For instance, if the vocabulary of melodic segments consisted solely of individual tones and the average vocabulary size was one, it would indicate that the model could not find any melodic units and segmented the melodies as individual tones. We refer to this measure as *avg_seg_len* and calculate it as the average length of all predicted segments across all chant melodies. This means we consider all duplicated melodic units, not just the unique ones, and therefore we do not calculate the average length of the melodic unit vocabulary but of all segments.

Melody Aligned with Words

Lacoste [2022] provides the text of the chants. Furthermore, as part of the data filtering and preparation by the Cornelissen et al. [2020b], word-based segments are also generated. Therefore, we can compare our segmentation with the natural word-based segmentation. However, the word-based segmentation may not necessarily be related to the potential correct segmentation. Hence, we could use a measure of similarity between the word-based segmentation and the predicted segmentation for analysis and discussion regarding the role of words in this task. The similarity score is calculated as follows:

$$mavw = 100 \cdot \frac{1}{W} \sum_{i=1}^{N} w_i. \tag{5.2}$$

The N is the number of chants, and w_i is the number of words in the text of chant with id i that end on the same tone as any of the segments. The value of W is the number of all words in the dataset. We have named this measure the *maww* score. Similarly to the previous case, note that shorter segments align more easily with words.

Melody Aligned with Phrases

As mentioned in Section 4.2, the dataset called GregoBase by Berten [2013] consists of annotated scores with pause marks. These pause marks indicate the ends of phrases where singers should take a breath. In the case of the melodic unit system, the melodic segments would likely align with these boundaries. We calculate the alignment of segments by phrases in a similar way to the maww score, except we align them with phrases. The aligned melody score is denoted as *mawp* and is calculated as:

$$mawp = 100 \cdot \frac{1}{P} \sum_{i=1}^{N} p_i, \tag{5.3}$$

where N is the number of chants, and p_i represents the number of phrases in the text of the chant with id i that end at the same time as any of the chant's segments. The value of P is the number of all phrases in the dataset.

Weighted Unique Final Pitch Count

In modern music theory, the final tone of a musical composition is determined by its musical scale. Similarly, in Gregorian chant, the final tone of the chant is related to its mode. A similar system may exist for potential melodic units. For this reason, we measure the number of unique final tones in the segments. Chants with fewer segments are likely to have a small number of unique final tones, so we consider those with more segments to be more relevant. Therefore, we take into account the number of unique final tones in each chant, weighted by the number of segments in the chant, and calculate the average across all segments of all chants. Let's denote the number of chants as N, the number of segments in the chant with id i as s_i, and the number of unique final tones in the segments of the chant with id i as f_i. Then, for $s = \sum_{i=1}^{N} s_i$, we calculate:

$$wufpc = \frac{1}{s} \cdot \sum_{i=1}^{N} (s_i \cdot f_i). \tag{5.4}$$

This result is referred to as the *wufpc* score. Note that in melodies with shorter segments, there are more segments. Therefore, when the average segment length is small, the probability of obtaining a higher wufpc value increases.

Vocabulary Levenshtein

Once we have a vocabulary, it is desirable for the segments to be as distinct as possible. Otherwise, the vocabulary might consist of two similar melodies with a small difference, which could be considered as the same unit in the system of melodic units. Therefore, we define the function *vocab_levenshtein* as follows:

$$vocab\_levenshtein = \frac{1}{|V|} \cdot \sum_{l \in L} \sum_{s \in S_l} (\frac{1}{l} \cdot med(LD_{l,s})), \tag{5.5}$$

where $|V|$ is the size of the vocabulary, L is the set of all possible segment lengths in the data, S_l is the set of all melodic units of length l, $med(X)$ denotes the median of a set of numbers X, and $LD_{l,s}$ represents the set of Levenshtein distances between the segment s and all other segments in the vocabulary of length l. If there is only one segment of length l, it is disregarded in the calculation.

5.3 Mode Scores

Modes can have a significant impact on segmentation. Each mode has its own characteristics, such as a different range of tones. Therefore, it is possible that each mode also has its own melodic unit vocabulary or a specific set of rules for assembling them. Of course, only if there is a melodic unit system.

5.3.1 Mode Classification

The task of mode classification is a problem where we aim to classify chants to their respective modes. Cornelissen et al. [2020b] addressed this task and found the approach based on segmentation as the most accurate one. They used simple melody segmentations, such as n-grams (all segments have the fixed length n) or segmentations by words or syllables. Therefore, we could measure the quality of the segmentation as its ability to predict modes as accurately as possible.

SVC Classification

Cornelissen et al. [2020b] utilized a TF-IDF vectorizer with a maximum of 5000 features to encode the segments. They then applied the SVC model for classification. In order to compare our segmentation with their proposals, we use the same architecture and the same set of hyperparameters. We denote the classification results obtained from that pipeline as *bacor_accuracy* and *bacor_f1*, as named after the primary author of Cornelissen et al. [2020b].

Naive Bayes Classification

As mentioned in Section 4.2, SVC performs well with complex and unstructured data. On the other hand, the performance of Naive Bayes is dependent on statistically independent features, so in the case of more complex feature interactions, Naive Bayes's performance is worse. Therefore, for classification, we also consider the Naive Bayes model to assess the simplicity and straightforwardness of the predicted segmentation. In this case, straightforwardness means that single melodic units in segmentation already indicate the modality of the whole melody well. It is not necessary to complexly combine features together to be able to predict the mode correctly. The Naive Bayes model uses features generated by the TF-IDF vectorizer with the same settings as the previous SVC model. And again, the goal is to achieve the most accurate prediction of chant modes. The resulting scores of the classification task using this pipeline are referred to as *nb_accuracy* and *nb_f1*.

5.3.2 Weighted Top Mode Frequency

For the purpose of analyzing the mode dependence, we are also interested in how individual melodic units are related to a specific mode. Firstly, we assign a mode to each melodic element in the vocabulary based on the chants that contain the element. This means that we go through all the chants and count the occurrences of each vocabulary item within chants of a particular mode. Then, we select the mode with the highest number of occurrences as the melodic unit mode. Let's denote the mode of melodic unit v as m_v. Secondly, we calculate the weighted average of the ratio of the number of the melodic unit in the assigned mode to the total number of occurrences of the melodic unit as follows:

$$wtmf = 100 \cdot \frac{1}{\sum_{v \in V} \sum_{m \in 1,2 \ldots,8} n_{v,m}} \sum_{v \in V} \frac{n_{v,m_v}}{\sum_{m \in 1,2 \ldots,8} n_{v,m}}. \tag{5.6}$$

Here, V represents the vocabulary of melodic elements, and $n_{v,m}$ is the number of occurrences of melodic unit v in all chants of mode m. This metric is referred to as *wtmf*.

5.3.3 Charts

In order to analyze the structure of the chant and the occurrence of segments, we visualized certain melody and segment properties in charts.

Density of Unique Segments Chart

Since each mode behaves differently, each may have its own characteristic melodic units. Therefore, we observe the segments of each mode that are present only in chants of that specific mode and not in any other chants of different modes. We refer to such melodic units as *unique melodic units* or *unique segments* of the respective mode. For each mode, we create a chart that visualizes the positions of these unique melodic units across all chants belonging to that mode. We call that chart as *density of unique segments*. The x-axis represents the percentage position within the melody, indicating whether it is at the melody's beginning, middle, or end. The y-axis represents the percentage of melodic segments that belong exclusively to that mode. We scale each melody into 400 bins. We then iterate through all the melodies of the mode and increment the value mapped to the bin by one if the bin corresponds to one of the unique melodic units. Each of these values is divided by the number of chants in that specific mode (which is also a number of segments corresponding to a particular bin), yielding the percentage of unique melodic units for each bin across all chants of the mode. Each bin represents 0.25 % of the chant, so the 200th bin represents the middle position. For instance, Figure 5.1 illustrates a situation where at the beginning of the chants, 16 % of the segments are used only in the particular mode, but by the end of the first fifth of the chants, it drops to just 8 % of unique melodic units. Towards the end of our example, chants rarely contain unique segments.

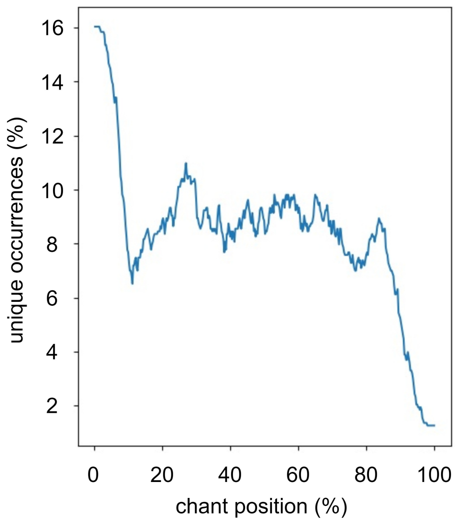


Figure 5.1: Example of the density of unique segments chart.

By analyzing this chart, we would be able to see how much the modes are related and how uniquely they behave. The chart will show whether the chant's beginning or end is more unique than the middle part, which they should be since they are linked with the mode-specific context chants. We could expect that there would not be many unique melodic units since the pair of authentic and plagal modes have many properties in common.

Average Segment Lengths Chart

We already have a metric for calculating the average segment length across all chants and their segments. However, we need to find out if the average segment length is the same for all chant intervals. Some positions may pose more challenges for segmentation than others. These problematic intervals could significantly affect the resulting average segment length score. Therefore, we scale all chants into 400 bins, similar to the previous chart construction. Instead of assigning a count to each bin, we assign a number that represents a sum of segment lengths that fall within that bin. To obtain an average, we divide each bin number by the total number of chants used. We calculate these averages separately for each mode. The resulting numbers are then visualized in a chart, where the x-axis represents the percentage position of the chants, while the y-axis represents the average segment length relative to the chant position. Figure 5.2 visualizes an example chart for chants segmentation, where the average segment length is nearly five at the beginning, is close to one in the middle, and is approximately three tones long at the end. Note that the avg_seg_len would always be lower than the average of all bin values since longer segments contribute to more bins than those short ones.

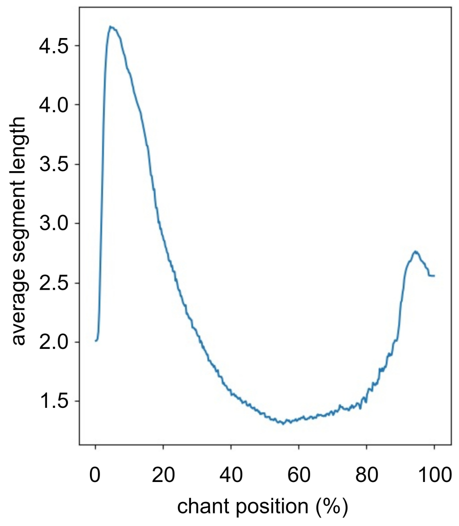


Figure 5.2: Example of the average segment length chart.

Segments Occurrences Chart

Information about the frequencies of melodic units could also be useful. All melodic units found during segmentation could be used approximately equally. But also, some melodic units may occur more frequently than others, especially when considering chants of a specific mode. Therefore, for each mode, we create a chart that visualizes the occurrence of individual melodic units. We sort the melodic units in a specific way. As in the wtmf, we select the mode with the highest number of occurrences among all chants belonging to that mode. Firstly, we sort the vocabulary elements by their modes, where the first group is assigned to mode 1, the second to mode 2, and so on. Secondly, we sort the melodic units in descending order based on their total number of occurrences across all chants of all modes. The chart consists of two lines. The gray line represents the total number of occurrences of melodic units in all chants. The blue line visualizes the occurrence count of melodic elements only in chants belonging to the mode for which the chart is being visualized. The example in Figure 5.3 consists of over 14000 melodic units, with the most frequent melodic unit being used more than 250 times. Below the chart, we also marked mode sections separated by the first filtration procedure.

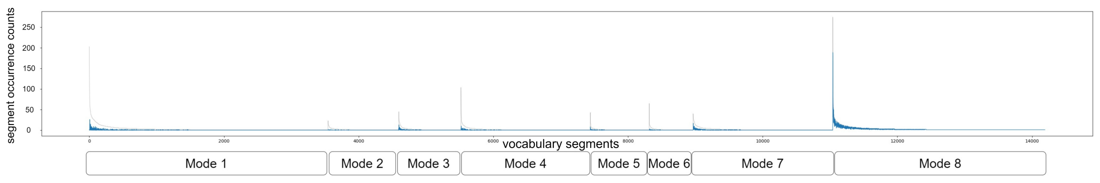


Figure 5.3: Example of the segments occurrences chart in chants of the mode 8.

In the case of the segments occurring frequently and approximately evenly in the segmentation, we can discuss a strict melodic unit system without memorization mistakes. Or there could be only segments that are not that frequent, indicating no melodic units repeated over different chants. In the third case, we could observe that some of the melodic units are frequent and the rest of them are rare, which could mean that many melodic units were memorized and transcribed wrongly, but also that there are melodic units that are used by many chants suggesting the melodic unit system existence. Another property of the chart describes how segments are shared among different modes. The blue line being placed only in the section of the respective mode would indicate that each mode behaves independently with a separate set of segments. But still, we would expect the blue line to be placed at least in the authentic-plagal mode pair similarly.

5.4 Feature Extraction

Extracting the strongest melodic segments from the model could also be necessary. They could be compared to other models and, later on, be analyzed in more detail. There are several ways to extract top melodic units. We are considering using the SVC model for mode classification, and feature extraction is performed based on the model's coefficients. While this approach could return features with substantial weights, it may also return rarely used segments that are not of particular interest to us. Instead, we could use a feature extraction approach based on an additive procedure. It works by starting with an empty set of features and then trying all possible features to expand the set, selecting the one with the highest SVC score in mode classification until we have a set of n features. However, this approach took a significant amount of computational time, and we could not finish it. Therefore, we will use feature extraction from the weight coefficients of the SVC model. To address the issue of extracting strong features that are rarely used, we will first extract 1000 features from the model and then select the 100 most frequently occurring ones. This way, we hope to extract 100 intense and frequent features. We refer to these segments as *top* 100 *melodic units*.

In the same way, as we defined the density of unique segments chart, we describe the *density of top segments* chart. We again scale all chants into 400 bins, and for each bin, we compute the number of top segments that lay on that bin. Then we calculate the percentage number of top segments considering all segments at each bin. Results are visualized in the chart as shown in Figure 5.4, where the x-axis represents the percentage position of chants, and the y-axis shows the percentage of top segments over all segments over all chants at the specific position. Each bin stands for 0.25 % of chant positions. The chart should imply the percentage of occurrences of the top features, so we can see if we can describe most segments using these features. But the chart is designed in a way that longer segments are counted more often in bins than the short ones. If top features contain mostly short segments, the chart would be more affected by those not included in the top features. However, we can still observe the position of top features, which could be expected to be more often at the beginning and end for linking with the mode-specific context chants.

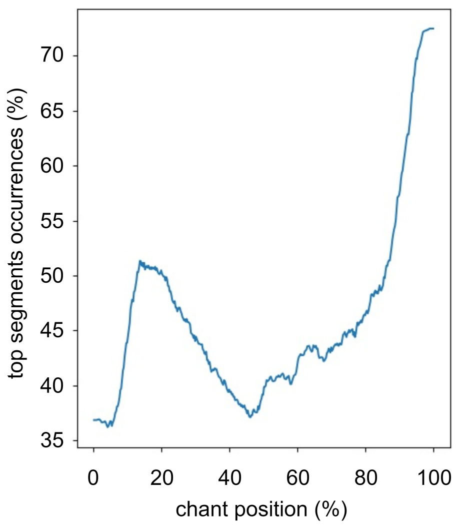


Figure 5.4: Example of the density of top segments chart.

Additionally, we observe the *top segments - modes* matrix that visualizes top segments distribution over modes. Each column stands for one of the top segments. Each row stands for one of eight modes. Each cell visualizes the occurrence of a particular segment in a given mode, with each column normalized to sum to one. Then, cells are colored by the default settings of the Python library by Hunter [2007]. The color spectrum starts with dark purple and ends with yellow, so a cell with yellow color means that the cell mode is the most dominant for that particular segment. The example of the top segments - modes matrix is shown in Figure 5.5. The matrix should analyze modes regarding the top features. It could happen that all cells are colored the same, but also, there could be one yellow cell in each column that would support that each mode behaves independently. But still, the authentic-plagal mode pair would probably be colored similarly.

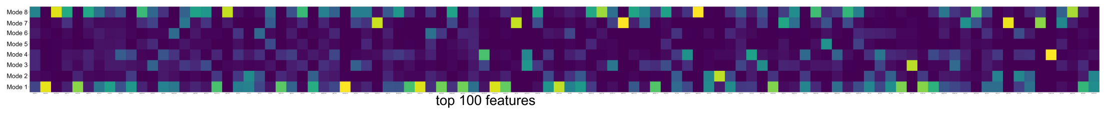


Figure 5.5: Example of the top segments - modes matrix.

6. Models

This chapter will introduce segmentation models that this work measures and compares. We will go through the structures of the models as well as the training and decoding processes. We will also propose an extension of the models, focusing on a more significant differentiation of the segments regarding modes. First, we will describe a naive, simple model based on unigrams of segments. Then, we will describe a model based on the Pitman-Yor process. Finally, we will discuss a model based on neural networks, specifically a model based on BERT. All of these models are implemented in the Python programming language.

6.1 Unigram Model

The unigram model consists of its vocabulary, the total number of segments, the count of occurrences of each segment, and the segment inverted index, which indicates all chants containing a particular melodic unit. The model's data structures are populated from the training dataset in an unsupervised manner during the training process. The model also has minimum and maximum segment size information as user-defined hyperparameters. We start with empty data structures. Then, we define the model's vocabulary of all possible unique segments in the training dataset, corresponding to predefined allowed segment sizes. This is necessary for the Laplacian smoothing we will use in training and decoding processes. Then, the model and its data structures are trained in several iterations, with the number of iterations determined by the user.

Formerly, let's define the probability of the corpus of chants C as:

$$p(C) = \prod_{c \in C} p(c). \tag{6.1}$$

We are looking for the chant segmentation $\bar{c} = s_1 ... s_m$ for each chant c in the corpus C. Each of segments $s_1, ..., s_m$ consists of the natural number of tones $t_1 ... t_o$. Compounding the chant segments $s_1, ..., s_m$ together gives back the chant c itself. Then let's denote the set of all segmented chants as $\bar{C} = \{\bar{c} : c \in C\}$. Our unigram model Θ consists of three sets of variables. The first one is the vocabulary V, the set of all allowed segments that occur in the training data. The second one is the set of variables $n_s = \#\{s | \exists \bar{c} \in \bar{C} : s \in \bar{c}\}$ for each segment $s \in \{s_c | \exists \bar{c} \in \bar{C} : s_c \in \bar{c}\}$. The last variable of the unigram model Θ is the $n = \sum_{s \in \{s_c | \exists \bar{c} \in \bar{C} : s_c \in \bar{c}\}} n_s$.

6.1.1 Training and Decoding

In order to train and decode the statistical model of the unsupervised segmentation task, let's describe the model as the HMM. Each hidden state corresponds to a segment, while observations are given by the melody tones. The number of hidden states is not fixed since we do not know the optimal number of segments of a particular chant, so we call it infinite HMM. Transition and emission distributions are then given by the model. In this section, we will describe the construction of the unigram model solving unsupervised segmentation as HMM.

First, we assign an initial segmentation to each chant. The unigram model offers two options for initial segmentation: segmentation based on words or random segmentation. Random segmentation is performed by iteratively sampling the length l of the next segment from a Gaussian distribution $l \sim \mathcal{N}(\mu, \sigma^2)$, where we set $\mu = 5$ and $\sigma = 2$. If the sampled length l does not satisfy the minimum and maximum segment size conditions, it is truncated to meet these requirements. Since there is no improvement with word-based initial segmentation, we will use random segmentation. Then, we will train the model using the Blocked Gibbs sampling algorithm, so as a first step, we will add the initial segmentation of all chants to the model.

Blocked Gibbs sampling is a variant of the general Gibbs sampling described in Section 2.1.1, where we sample a block of variables at a time. In this case, the block is the chant with all its segments, so we sample all chant segments in one sampling step. Blocked Gibbs sampling removes the segmented melody from the model, predicts a new melody segmentation based on all other segmented melodies, and then adds the new melody segmentation back into the model. This process is done iteratively for all chants, and the order of the chants is different and random each time. But first, we still need to add all the initial segments to the model.

As part of the procedure for adding a new segmented chant to the model, we need to increase the total number of segments by the number of newly predicted chant segments. Then, for each segment in the segmentation, we need to increment the count of segment occurrences by one. Finally, we also need to maintain the information that these segments are included in this particular chant. Therefore, we must add the chant id to all segments of this chant in the segment inverted index list.

On the other hand, the removal procedure takes the old chant segmentation and removes all records related to the chant from the model's data structures. In contrast to the previous function, here we need to decrement the total number of segments by the number of segments in this segmented chant. We also need to iterate over all chant segments and decrease the count of segment occurrences for each. Additionally, we need to remove the chant id from the segment inverted index for each segment.

As part of the training process, we need to segment the chant based on all other chants and their segmentations. The probability of segment s using Laplace smoothing is denoted as:

$$p(s|\Theta) = \frac{n_s + \alpha}{n + \alpha \cdot |V|}, \tag{6.2}$$

where Θ represents the parameters and variables of our model, n is the current total number of segments, n_s is the count of segment occurrences for segment s across all other chants, and $|V|$ is the size of the vocabulary of all possible allowed segmentations. The parameter α is a hyperparameter set to $\alpha = 1e-14$. Ideally, we want to maximize $p(\bar{c}|\Theta)$ for the segmentation $\bar{c}$ of the chant c. We can rewrite it as $\bar{c} = s_{c1}, s_{c2}, ..., s_{cm}$, with m consecutive segments of the chant c. Then we have:

$$p(\bar{c}|\Theta) = \prod_{i=1}^{m} p(s_{ci}|\Theta). \tag{6.3}$$

We construct a trellis, a graph of possible paths of segmentations and their probabilities. The graph considers only segments that satisfy the minimum and maximum segment size conditions. For decoding the trellis, we use the Viterbi algorithm. At each node, we consider the top k paths to that node. The final node consists of the k most probable segmentation paths of the entire chant from start to end. During training, if we segment the chant, we select one of the top k segmentations proportional to the probability distribution of these segmentations, normalized so that their sum equals one. This means the probability of selecting segmentation $\bar{c}$ is its probability $p(\bar{c})$ divided by the sum of probabilities of all top k segmentations. On the other hand, during decoding, we do not sample but instead select the most probable segmentation from the top k final segmentation paths. This means we choose the segmentation with the highest probability from the best k final segmentations. In our experiments, we set $k = 15$.

In the case of very small segment probabilities, all the final k segmentation probabilities could be rounded to zero, resulting in a uniform distribution and incorrect sampling. Additionally, we could not select the most probable segmentation path. Therefore, we compute the logarithms of these probabilities during the training and decoding processes.

6.1.2 Mode Extension

Let's consider that each mode has its own rules for compounding segments and its own vocabulary. This would mean each mode has a different set of characteristic melodic units. If this were true, it would make sense to create a vocabulary of melodic units for each mode separately. Therefore, we also propose an approach that extends from the previous model to a model that includes eight vocabularies, eight total chant counts, etc. Each mode has its own set of data structures. At the beginning of the training, we divide our training dataset into eight subsets, each containing chants of only one particular mode. Then we train all the data structures separately without interdependence.

We know the mode of the chant during training. However, during testing and decoding, we don't have such information. Therefore, we first need to predict the mode and then predict the segmentation using parameters associated with the chosen mode. When predicting the mode within the model, we want to choose a mode m that maximizes:

$$p(m|\bar{c}) = \frac{p(\bar{c}|m) \cdot p(m)}{p(\bar{c})} \propto p(\bar{c}|m) \cdot p(m) \propto p(\bar{c}|m). \tag{6.4}$$

This equation follows from Bayes' theorem, where $p(\bar{c})$ is the same value for all modes and $p(m) = \frac{1}{8}$ is also the same because it is a uniform distribution. Therefore, we consider all modes of the chant and predict the most probable chant segmentation in all eight cases. We have eight different probabilities and eight different chant segmentations. Then we compare these probabilities and select the mode m_c with the most probable segmentation $\bar{c}_{m_c}$. Then the $p(m_c|\bar{c}_{m_c})$ is maximized. If we would consider the $\bar{c}_{m_c}$ to be segmented in the context of another mode, it could not be more probable than the $p(m_c|\bar{c}_{m_c})$ since it is more probable than the most probable segmentation considering another mode. And also, there is no more probable chant segmentation considering the mode m_c since the $\bar{c}_{m_c}$ was chosen as the most probable one.

The probability that the chant c will be segmented as $\bar{c}$ is then calculated as follows:

$$p(\bar{c}|\theta) = \sum_{m \in \{1,\ldots,8\}} p(\bar{c}|m), \tag{6.5}$$

where each of the $p(\bar{c}|m)$ is computed as the product of segment probabilities associated with mode m.

6.2 Nested Hierarchical Pitman-Yor Language Model

In this section, we will introduce the model described by articles Mochihashi et al. [2009], Teh [2006b], Mochihashi and Sumita [2007], and Teh [2006a]. This work includes the implementation of NHPYLM in the Cython language. Cython is a programming language similar to Python but supports C data types. The Cython compiler compiles the code into C and wraps it into an interface so the generated module can then be easily called and used in Python, just like any other Python package or module (Behnel et al. [2011]). The main advantage here is that Cython significantly reduces the overhead of Python, resulting in faster code execution. Otherwise, if we implemented it in Python, we would be extremely limited in training and debugging the NHPYLM model.

6.2.1 Model Structure

As we know from the statistical background provided in Section 2.1.1, we have a distribution G generated from the Pitman-Yor process:

$$G \sim PY(d, \theta, G_0), \tag{6.6}$$

where d is the discount factor, θ is the concentration, and G_0 is the base measure. We generate a distribution G that resembles the base distribution G_0 according to a given similarity measure θ. Let's consider the distribution of unigram segments and denote it as $G_1 = \{p(\cdot)\}$. Similarly, let's assume the distribution of bigram segments conditioned on a context segment s and denote it as $G_2 = \{p(\cdot|s)\}$. Both distributions have many similarities. Therefore, we could generate G_2 from the Pitman-Yor process using G_1 as the base measure:

$$G_2 \sim PY(d, \theta, G_1). \tag{6.7}$$

This way, we obtain a *hierarchical* model with the base distribution of unigram segments G_1, which is shared among many distributions of bigram segments given the context segment G_2. Each possible context segment has its distribution. Let's call this hierarchical structure the segment hierarchical Pitman-Yor language model (SHPYLM). The tree-like structure is composed of nodes. In analogy with a Chinese restaurant, each node contains a dictionary of unique dish keys and values as a set of tables where the particular dish is served. Each table is represented by a positive number of customers sitting at the table. In the analogy, the dish represents the segment we add, remove or observe. Furthermore, we keep track of the total number of tables and customers, as well as the number of customers for each dish. In the case of bigram nodes (the tree layer of depth two), we also store the context, which is the previous segment of the dish. Then, we denote $c(h)$ as the total number of customers in the node described by context h, $c(s|h)$ as the total number of customers in the node of dish segment s, t_{hs} as the number of tables in the node of the context h associated with dish s, and t_h as the total number of tables in the node. The final probability of dish segment s given the context segment h is then calculated as follows:

$$p(s|h) = \frac{c(s|h) - d \cdot t_{hs}}{\theta + c(h)} + \frac{\theta + d \cdot t_h}{\theta + c(h)} \cdot p(s|h^{'}), \tag{6.8}$$

where $h^{'}$ refers to the shorter context, in our case, the higher layer of the hierarchical tree structure. Basically, $p(s|h^{'})$ is the probability computed from the base measure of the current Pitman-Yor process.

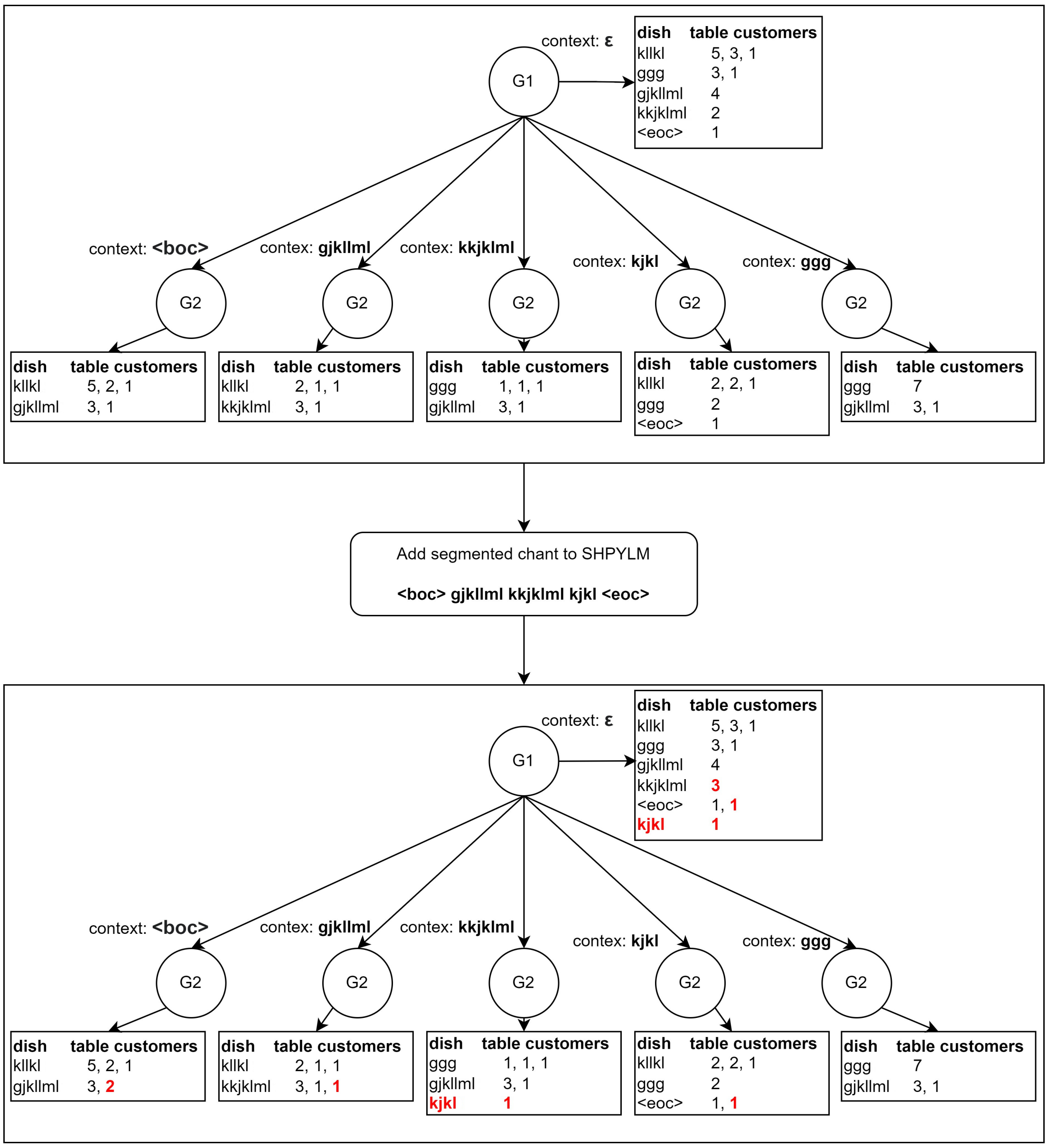
However, we still do not have the base measure G_0 for the distribution G_1. Theoretically, we could use the inverse size of the vocabulary as a uniform distribution of segments if the segment vocabulary were finite. Since we do not know the segment vocabulary, we would have to consider all possible combinations of tones, which would give us a countably infinite vocabulary size. Therefore, instead of that, we will use the tone hierarchical Pitman-Yor language model (THPYLM) as the base measure G_0, the nested HPYLM to the SHPYLM. In this case, the THPYLM returns the tone ∞-gram model distribution, so its tree hierarchical structure does not have a defined maximal depth. On the other hand, the SHPYLM's final layer contains distributions of bigrams, so its depth is set to two. The base measure for the THPYLM is the number of all possible tones $|T|$, which is known and finite. The probability of a segment $s = t_1 t_2 ... t_k$ with a length of k and tones $t_1 ... t_k$ given predicted from THPYLM is calculated as follows:

$$p(t_1 ... t_k) = \prod_{i=1}^{k} p(t_i | t_1 ... t_{i-1}). \tag{6.9}$$

Now we almost have the G_0 distribution. Each node of the tree structure of THPYLM stores the same information as the nodes of SHPYLM. However, the dish corresponds only to the tone, not the segment. Additionally, each node contains information about stops and passes. The number of stops in a node identifies the number of customers added to the tree with a context tone length equal to the depth of the node. The number of passes is the count of passes through the node when we use this node to get to the target node of the given context when we start in the root. We will describe the process in more detail in the following section. Let's denote the context $h = t_1 ... t_{i-1}$ for the tone $t = t_i$. Then, according to Mochihashi and Sumita [2007], since we have an infinite model, we compute the probability $p(t_i | t_1 ... t_{i-1})$ as:

$$p(t|h) = \sum_{n=0}^{\infty} p(t|h, n) \cdot p(n|h). \tag{6.10}$$

The value $p(t|h, n)$ is computed from HPYLM as in Equation 6.8, where the dish is a tone, and n indicates the depth, i.e., the order of the node (which represents the size of the context). The probability distribution $p(n|h)$ represents the probability that the context h has a latent Markov order of n. If we consider



dish	table customers
kllkl	5, 3, 1
ggg	3, 1
gjkllml	4
kkjklml	2
<eoc>	1

dish	table customers
kllkl	5, 2, 1
gjkllml	3, 1

dish	table customers
kllkl	2, 1, 1
kkjklml	3, 1

dish	table customers
ggg	1, 1, 1
gjkllml	3, 1

dish	table customers
kllkl	2, 2, 1
ggg	2
<eoc>	1

dish	table customers
ggg	7
gjkllml	3, 1

Add segmented chant to SHPYLM

<boc> gjkllml kkjklml kjkl <eoc>

dish	table customers
kllkl	5, 3, 1
ggg	3, 1
gjkllml	4
kkjklml	3
<eoc>	1, 1
kjkl	1

dish	table customers
kllkl	5, 2, 1
gjkllml	3, 2

dish	table customers
kllkl	2, 1, 1
kkjklml	3, 1, 1

dish	table customers
ggg	1, 1, 1
gjkllml	3, 1
kjkl	1

dish	table customers
kllkl	2, 2, 1
ggg	2
<eoc>	1, 1

dish	table customers
ggg	7
gjkllml	3, 1

Figure 6.1: Example of the process of adding segmented chant "*<boc> gjkllml kkjklml kjkl <eoc>*" encoded using the Volpiano note symbols into SHPYLM. Changed values of the SHPYLM are colored red. Note that the sum of dish customers in $G1$ distribution corresponds to the number of tables over all $G2$ distributions. When adding the segment to the node, the dish table is picked or created new randomly proportionally to the corresponding probability distribution.

the path from the THPYLM root to the node described by the context h, for the node of order n on this path, we compute:

$$p(n|h) = \frac{a_n + \alpha}{a_n + b_n + \alpha + \beta} \cdot \prod_{i=0}^{n-1} \frac{b_i + \beta}{a_i + b_i + \alpha + \beta}, \tag{6.11}$$

where a_k represents the number of stops in the node of order k that is part of the context's node path from the root, b_k represents the number of passes through the same node, and α and β are hyperparameters. We set them to $\alpha = 1$ and $\beta = 1$. Since we cannot compute the infinite recursion, we stop the computation at order n when $p(n|h) < \epsilon$, where in our case, $\epsilon = 1e - 12$.

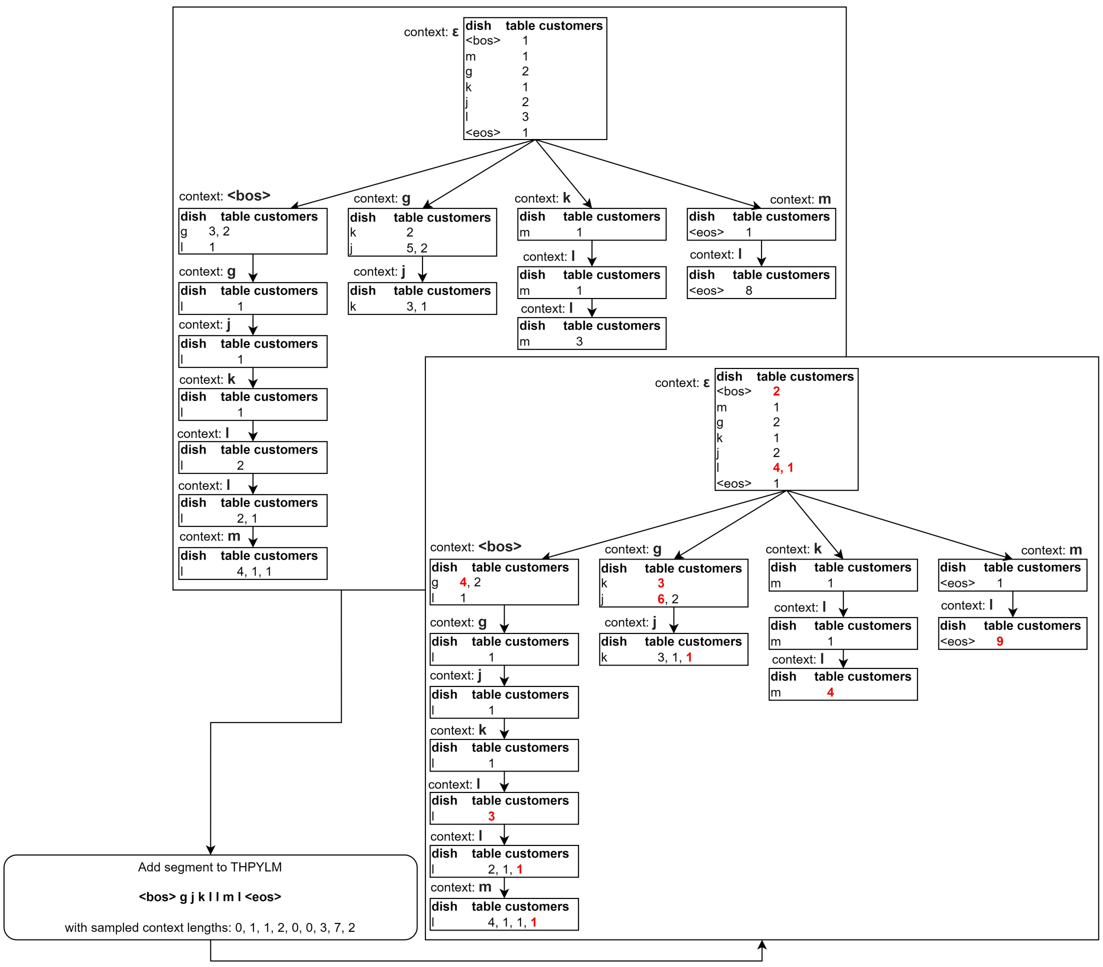


Figure 6.2: Example of the process of adding segment "*gjkllml*" encoded using the Volpiano note symbols into THPYLM. Changed values of the THPYLM are colored red. The context lengths are not fixed to 1 as in the case of SHPYLM. Context lengths are sampled first, so according to our example, we add $<bos>$ with empty context, g with context $<bos>$, j with context g, k with context gj, etc. The last tone's context length was even sampled to be 7, so the last l tone has context $<bos>gjkllm$. When adding the tone to the node, a dish table is picked or created new randomly proportionally to the corresponding probability distribution.

The probability distribution of a segment s given by the THPYLM, as described in Equation 6.9, assigns lower probabilities to longer segments. This means the model will prefer shorter segments, especially those containing only one tone. To address this issue, Mochihashi et al. [2009] introduced the Poisson correction:

$$p(t_1...t_k) = p(t_1...t_k, k) = \frac{p(t_1...t_k, k|\Theta)}{p(k|\Theta)} \cdot Po(k|\lambda) \tag{6.12}$$

for a segment $t_1...t_k$ with k tones. First, we create a probability distribution from the THPYLM model Θ as $p(t_1...t_k, k|\Theta)$. Then we divide this probability by the distribution of the model Θ predicting a segment of length k. Essentially, we suppress segment lengths that are more probable according to the THPYLM model. We then use the Poisson distribution:

$$Po(k|\lambda) = e^{-\lambda}\frac{\lambda^k}{k!}. \tag{6.13}$$

The hyperparameter λ and the precomputation of $p(k|\Theta)$ are described in detail in Section 6.2.4 focused on the model's hyperparameters.

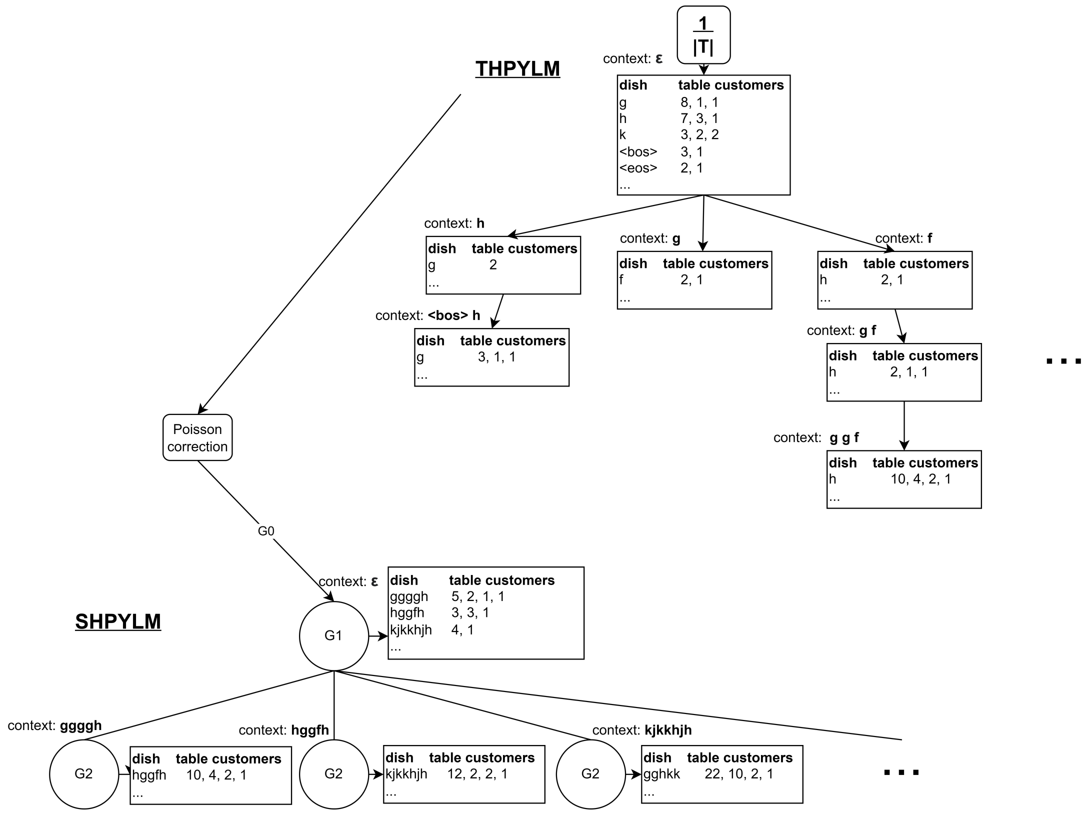


Figure 6.3: Ilustration of the NHPYLM nodes related to the chant segmentation "*ggggh hggfh kjkkhjh gghkk kkjkhg*" encoded using the Volpiano note symbols. The THPYLM shows the nodes related to the *hggfh* segment (with the $< bos >$ and $< eos >$ tokens) with sampled context lengths $\{0, 0, 2, 0, 1, 3, 0\}$.

6.2.2 Adding and Removing of Customers

We have already defined the structure of the NHPYLM model, which consists of THPYLM, SHPYLM, and the Poisson correction. The model predicts probabilities of bigram segments based on the training chants added to the model. Let's assume we have a segmented training chant. Then we need to add individual bigram segments to the model. Using the analogy of a Chinese restaurant, a segment represents a customer, and we are looking for a segment's table, or eventually, we create a new one. First, we need to find the SHPYLM node that represents the first gram of the bigram (the context) and potentially create a node if it does not exist. Then we consider the second gram, our segment, as a dish to which we want to add a new customer. We randomly select an existing table k for dish s with context h in proportion to:

$$max(0, c_{hsk} - d), \tag{6.14}$$

where c_{hsk} is the number of customers at table k for dish s in the context node h, and d is the discount factor. Or we create a new table with probability proportional to:

$$(\theta + d \cdot t_{h\cdot}) \cdot p(s|h^{'}), \tag{6.15}$$

where θ is the concentration parameter, $t_{h\cdot}$ is the total number of tables in the context node h ($\sum_s t_{hs}$), and $p(s|h^{'})$ is the probability of dish s given the shorter context, which is based on the base measure distribution. When selecting an existing table, we increase the number of customers at the table, the total number of customers for that dish, and the total number of customers in the SHPYLM node by one. In the case of creating a new table, we follow the same procedure, but additionally, we increase the number of tables for the dish in the node and add a new customer for the added table in the parent node for the same segment represented as the dish, but without the context. We follow the same procedure as in the bigram node, only in the unigram node. In the case of creating a new table in the unigram node, we add the segment s to the THPYLM.

In the case of SHPYLM, we add all the segments of the segmented chant. Similarly, we need to proceed when adding a segment to THPYLM. In THPYLM, we need to add all the segment tones. In the segment version, we always add bigrams since SHPYLM has a fixed depth. This means the context is always another segment preceding the target segment. On the other hand, THPYLM represents an ∞-gram model, so the tone context should be handled differently. First, for each tone, we sample the length of the context, and then we add each segment's tone with its context to the model, similar to the case of SHPYLM. When sampling the context length, we consider all possible context lengths: zero, the number of tones in the segment before the target tone, or any natural number in that interval. Then, we randomly select a length according to the distribution proportional to:

$$p(t_i|t_{i-n}...t_{i-1}) \cdot p(n|t_{i-n}...t_{i-1}), \tag{6.16}$$

for the context length n. The probability $p(t_i|t_{i-n}...t_{i-1})$ is computed similarly to Equation 6.8, where the dish is the tone t_i. The second probability $p(n|t_{i-n}...t_{i-1})$ corresponds to Equation 6.11, where $h = t_{i-n}...t_{i-1}$. We store all the sampled context lengths of all tones for a specific segment, which we would use in reverse to remove this segment from THPYLM. After sampling the lengths, for each tone, we find the node corresponding to the context (the depth of the node is the same as the context length), and then we add the tone to the node in the same way as we would add a segment to SHPYLM. We only increase the number of stops in the current node and increment the number of passes of all parent nodes up to the root by one. When creating a new table, we need to add a customer for the tone to the parent node again. We need to add new customers to the parent nodes as long as we have not reached the root or until no new table is created. However, when adding new customers due to a new table in any child nodes, we do not update the number of passes or stops again.

Removing segments from the model follows the reverse process. We take a bigram, find its corresponding node in SHPYLM, and remove one customer from the table of segment dish. The table is randomly selected proportional to c_{hsk}, the number of customers at table k of dish s in context node h. If the table becomes empty, we need to remove it, and we also need to remove the customer from the parent node of the same dish segment. However, the node's total number of customers, dish customers, and table customers should be decreased by one. If the table is removed, the number of dish tables and the number of the total node's tables must be decreased by one. If we were to remove a table from the root node, we would need to remove the segment from THPYLM. We take the sampled context lengths generated during the adding procedure for a specific segment and remove from THPYLM all the segment tones with these context lengths. We need to remove the tone customers from the nodes corresponding to the context, so we randomly select the k-th table of tone t and context h proportional to c_{htk} and decrease all the counts as in the case of removing a segment from SHPYLM. Furthermore, we decrease the number of stops in the node of the given context by one, and all the counts of passes of all parent nodes would be decreased as well. In the case of an empty table, we need to remove the customer from the parent node until there is no parent node left or there is no removed table. We also need to remove information about that table from the node itself. Passes and stops are only updated when removing the customer from its original node for the first time, not when removing customers from the parent nodes due to the removed tables.

6.2.3 Training and Decoding

Similar to the unigram model, the blocked Gibbs sampler, described in Figure 6.4, is also used to train the NHPYLM model. First, we need to add all initial segmentations of all training chants to the NHPYLM model. The initial segmentation assumes that each chant has only one segment, which is the chant melody itself. Then we perform J iterations. In each iteration, we take a random permutation of chants and, in this order, for each chant, we first remove its segmentation from the model, then we sample a new segmentation, and finally, we add the new segmentation back to the model. The only question now is how to sample a new segmentation based on the current state of the NHPYLM model.

Mochihashi et al. [2009] described the Forward-Backward algorithm, which can also be used for sampling new segments of a chant. The algorithm consists of two parts: Forward filtering and Backward sampling. Given a sequence of tones $t_1 ... t_n$, the Forward filtering part precomputes a probability array $\alpha[n][k]$ for the chant, representing the probability that the final subsequence $t_{n-k+1}t_{n-k+2}...t_n$ is an independent melodic unit. We then marginalize the previous segmentations and calculate the probability using the recursive equation:

$$\alpha[n][k] = \sum_{j=1}^{n-k} p(t_{n-k+1}^{n} | t_{n-k-j+1}^{n-k}) \cdot \alpha[n-k][j], \tag{6.17}$$

where t_a^b denotes a subsequence starting at index a and ending at index b as $t_a t_{a+1} ... t_b$. The recursion starts with the value $\alpha[0][0] = 1$. Equation 6.17 assumes that the possible segment length could be any positive number. Due to time

```
1:  for $j = 1 \cdots J$ do
2:      for $s$ in randperm $(s_1, \cdots, s_D)$ do
3:          if $j > 1$ then
4:              Remove customers of $\mathbf{w}(s)$ from $\Theta$
5:          end if
6:          Draw $\boldsymbol{w}(s)$ according to $p(\mathbf{w}|s, \Theta)$
7:          Add customers of $\mathbf{w}(s)$ to $\Theta$
8:      end for
9:      Sample hyperparameters of $\Theta$
10: end for
```

Figure 6.4: Blocked Gibbs sampler of word segmentation task (Mochihashi et al. [2009]).

complexity considerations, we need to limit the maximum segment length to a number $L = 7$. The equation would then look like this:

$$\alpha[n][k] = \sum_{j=1}^{min(L,n-k)} p(t_{n-k+1}^{n}|t_{n-k-j+1}^{n-k}) \cdot \alpha[n-k][j], \tag{6.18}$$

where $k \in \{1, 2, ..., L\}$. We use these probabilities in the Backward sampling part to sample a new segmentation from the chant's end. We consider the symbol <*eoc*> as the end-of-chant symbol. Let's denote the number of tones in the chant as N. Then, we randomly select a length for the final segment $k \in \{1, 2, ..., L\}$ proportional to:

$$\alpha[N][k] \cdot p(< eoc > |t_{N-k+1}^{N}). \tag{6.19}$$

The probability of the bigram $p(s_i|s_{i-1})$ is calculated from the NHPYLM model using Equation 6.8. Once we have sampled the length k_s of the last segment, we also have the final segment $s = t_{N-k+1}^{N}$. We can then further sample randomly according to the value:

$$\alpha[n][k] \cdot p(s|t_{n-k+1}^{n}) \tag{6.20}$$

for $n = N - k_s$. This selects a sample for the second segment, the predecessor of the final segment. Equation 6.20 can be generalized to sample all other segment lengths based on the successor segment until we reach the beginning of the chant.

To decode the segmentation of a given chant, we use the Viterbi algorithm, so we precompute the array $\alpha[n][k]$ in the same way as in Equation 6.17. Then, we backwardly decode the chant's segmentation. Again, we start with the $< eoc >$ symbol. We consider all allowed segment lengths and select the length k that returns the highest probability in Equation 6.19. Instead of random selection, we use the *argmax* function. Then, according to Equation 6.20, we search for the length of successively for all segments that returns the highest probability until we reach the beginning of the chant. The probability of the chant segmentation $\bar{c} =< boc > s_{c1}, s_{c2}, ..., s_{cm} < eoc >$ predicted by the NHPYLM model Θ is then calculated as:

$$p(\bar{c}|\Theta) = p(s_1| < boc >) \cdot p(< eoc > |s_m) \prod_{i=2}^{m} p(s_i|s_{i-1}) \tag{6.21}$$

```
1: for $n = 1$ to $N$ do
2:     for $k = \max(1, n - L)$ to $n$ do
3:         Compute $\alpha[n][k]$ according to Equation 6.17
4:     end for
5: end for
6: Initialize $n \leftarrow N$, $i \leftarrow 0$, $s_0 \leftarrow < eoc >$
7: while $n > 0$ do
8:     Draw $k \propto p(s_i|c_{n-k+1}^{n}, \Theta) \cdot \alpha[n][k]$
9:     Set $s_i \leftarrow t_{n-k+1}^{n}$
10:    Set $n \leftarrow n - k$, $igetsi + 1$
11: end while
12: Return $\bar{c} = s_i, s_{i-1}, \cdots, s_1$.
```

Figure 6.5: Forward-Backward sampling algorithm looking for a segmentation $\bar{c}$ of chant c (Mochihashi et al. [2009]).

for the beginning symbol of the chant $< boc >$ and the end symbol of the chant $< eoc >$. The probability of the bigram $p(s_i|s_{i-1})$ is calculated from the NHPYLM model using Equation 6.8.

In Python, small probabilities are rounded to zero. However, preserving information about them is important for the sampling and the segmentation comparison. We need to scale the array $\alpha[n][k]$ to prevent underflow. The scaled array $\alpha^{'}[n][k]$ is defined as follows:

$$\begin{aligned}\alpha^{'}[n][k] =& \frac{a[n][k]}{\sum_{k_{nrm}=1}^{min(L,n)} a[n][k_{nrm}]}\\ a[n][k] =& \Big(\sum_{j=1}^{min(L,n-k)} p(t_{n-k+1}^{n}|t_{n-k-j+1}^{n-k}) \cdot \alpha^{'}[n-k][j]\Big)\cdot\\ & \Big(\prod_{k_{scl}=2}^{k} scale[n-k_{scl}+1]\Big)\\ scale[n] =& \sum_{k_{nrm}=1}^{min(L,n)} a[n][k_{nrm}].\end{aligned} \tag{6.22}$$

We can rewrite the scaled $\alpha^{'}[n][k]$ as:

$$\begin{aligned}\alpha^{'}[n][k] &= \frac{\alpha[n][k] \cdot \prod_{i=1}^{n-1} scale[i]}{\Big(\sum_{k_{nrm}=1}^{min(L,n)} \alpha[n][k_{nrm}]\Big)\Big(\prod_{i=1}^{n-1} scale[i]\Big)}\\ &= \frac{\alpha[n][k]}{\sum_{k_{nrm}=1}^{min(L,n)} \alpha[n][k_{nrm}]}.\end{aligned} \tag{6.23}$$

The advantage of scaling is that we work with high values of probabilities throughout the training and decoding processes. We also keep all necessary information because the proportional relationship remains the same when sampling the previous segment length ending at the n-th position since the scaled $\alpha^{'}[n][k]$ is normalized over all possible lengths k. Therefore, we use the $\alpha^{'}[n][k]$ in the training and decoding processes instead of the $\alpha[n][k]$.

6.2.4 Hyperparameters

The NHPYLM model we described consists of many hyperparameters. Among them are those that users choose as fixed values, such as the maximum segment size L set to $L = 7$, the number of training iterations, or the coefficients for calculating pass and stop probabilities, α, and β. On the other hand, some hyperparameters are learned during training and adapted to the training data. The first is λ for Poisson correction, and the second is the discount factor d with concentration θ. These hyperparameters are initialized at the beginning by user and then updated after each training iteration, as shown in Figure 6.4.

The coefficient λ is chosen from a Gamma distribution $\lambda \propto \text{Gamma}(\alpha_\lambda, \beta_\lambda)$. Initially, we set $\alpha_\lambda = 6.0$ and $\beta_\lambda = 1.2$. The selected λ corresponds to the probability distribution X shown in Figure 6.6. With this setting, the Poisson

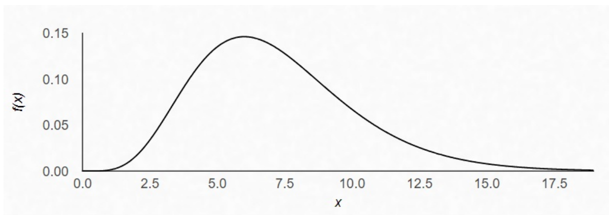


Figure 6.6: Probability distribution Gamma($\alpha = 6.0, \beta = 1.2$) (Bognar [2021]).

distribution based on sampled λ favors longer segments. However, λ is updated at each training iteration. We consider the dictionary of generated melodic units V seen in segmented chants. Then, λ is sampled from the posterior distribution

$$\begin{aligned} p(\lambda|V) &\propto p(V|\lambda) \cdot p(\lambda) \\ &= \text{Gamma}(\alpha_\lambda + \sum_{s \in V} t_s \cdot |s|, \beta_\lambda + \sum_{s \in V} t_s), \end{aligned} \tag{6.24}$$

where α_λ and β_λ are the initial hyperparameters, t_s is the count of segment s in the root unigram node, and $|s|$ is the length of segment s. As part of the Poisson correction, we also use the distribution $p(k|\Theta)$, which indicates the probability that the model Θ generates a segment of length k. Initially, we use a uniform distribution for all lengths k to have equal probabilities. After each training iteration, we update this distribution by sampling 20,000 segments from the THPYLM model. We start from the beginning-of-segment symbol $< bos >$ and consider all tones, calculating their probabilities with respect to the THPYLM model with the given context. The new tone of the segment is sampled proportionally to these probabilities. We expand the context with the sampled tone and sample a new one. We continue this process until we sample the end-of-segment symbol $< eos >$ or until the generated segment reaches the length $L = 7$. Then, the distribution of segment length k is calculated as follows:

$$p(k|\Theta) = \frac{c_k + 1}{c + L}, \tag{6.25}$$

where c_k is the count of sampled segments of length k, and c is the total count of sampled segments (in this case, $c = 20,000$). We also use Laplace smoothing in this calculation.

Other hyperparameters updated after each training iteration are d and θ. Each layer of THPYLM and SHPYLM has its own hyperparameter values d and θ. Initially, we assign them all the same starting values, set as $d = 0.5$ and $\theta = 2.0$. Then, we denote $\mathbf{u}$ as the context, which can be empty or consist of one segment in the case of SHPYLM, or it can be empty or contain any natural number of tones considering the THPYLM. As described by Teh [2006a], we need to precompute $x_{\mathbf{u}}$, $y_{\mathbf{u}i}$, and $z_{\mathbf{u}skj}$ as follows:

$$x_{\mathbf{u}} \sim \text{Beta}(\theta_{|\mathbf{u}|} + 1, c_{\mathbf{u}\cdot\cdot} - 1) \tag{6.26}$$

$$y_{\mathbf{u}i} \sim \text{Bernoulli}\Big(\frac{\theta_{|\mathbf{u}|}}{\theta_{|\mathbf{u}|} + d_{|\mathbf{u}|} i}\Big) \tag{6.27}$$

$$z_{\mathbf{u}skj} \sim \text{Bernoulli}\Big(\frac{j-1}{j - d_{|\mathbf{u}|}}\Big) \tag{6.28}$$

for θ and d given the length of the context, such as the layer depth $|\mathbf{u}|$, the count of customers $c_{\mathbf{u}\cdot\cdot}$ in the node of the context $\mathbf{u}$, and for each natural number i in the range $1..t_{\mathbf{u}\cdot}$, where $t_{\mathbf{u}\cdot}$ is the count of all tables in the node of the context $\mathbf{u}$. $z_{\mathbf{u}skj}$ is calculated for each customer j of each table of the specific dish s in the node of the given context $\mathbf{u}$. Then, for each layer at depth m, we sample new values for d_m and θ_m:

$$\begin{aligned} d_m \sim \text{Beta}\Big(a_m + \sum_{\mathbf{u}:|\mathbf{u}|=m, t_{\mathbf{u}\cdot} \geq 2} \sum_{i=1}^{t_{\mathbf{u}\cdot}-1} (1 - y_{\mathbf{u}i}), \\ b_m + \sum_{\mathbf{u},s,k:|\mathbf{u}|=m, c_{\mathbf{u}sk} \geq 2} \sum_{j=1}^{c_{\mathbf{u}sk}-1} (1 - z_{\mathbf{u}skj})\Big) \end{aligned} \tag{6.29}$$

$$\theta_m \sim \text{Gamma}\Big(\alpha_m + \sum_{\mathbf{u}:|\mathbf{u}|=m, t_{\mathbf{u}\cdot} \geq 2} \sum_{i=1}^{t_{\mathbf{u}\cdot}-1} y_{\mathbf{u}i}, \beta_m - \sum_{\mathbf{u}:|\mathbf{u}|=m, t_{\mathbf{u}\cdot} \geq 2} \log x_{\mathbf{u}}\Big) \tag{6.30}$$

for fixed hyperparameters $a_m, b_m, \alpha_m, \beta_m$, which we set to $a_m = 1, b_m = 1, \alpha_m = 1, \beta_m = 1$. First, we select θs and ds for SHPYLM, and then we follow the same procedure to select θs and ds for THPYLM. In Equation 6.29, $c_{\mathbf{u}sk}$ denotes the count of customers in the node of the context $\mathbf{u}$ for dish s and table k.

6.2.5 Mode Extension

Just as we described the extension of the Unigram model based on chant modes in Section 6.1.2, we propose the same extension for the NHPYLM model. Therefore, we create eight independent NHPYLM models and wrap them into a single NHPYLM Modes model, where each NHPYLM model is mapped to one of the modes. Then we split all training data into eight datasets, each consisting only of chants belonging to its corresponding mode. We will then train each NHPYLM separately. When segmenting an unseen chant from the validation or test dataset, we first need to predict the mode and then segment the chant using the NHPYLM model of that mode. We choose the mode m for chant c to maximize Equation 6.4 in the same way as in the case of the unigram model modes in the Section 6.1.2. The probability of segmenting a chant generated by the model is then equivalent to Equation 6.5, where the probability of segmenting a chant by one of the eight NHPYLM models is calculated according to Equation 6.21.

6.3 BERT

Another model that we will address is the model proposed by Li et al. [2022] based on BERT. They used this model for unsupervised word segmentation. Since they provided the model's source code in Python, we will build upon it. We will only adapt the model to our specific task.

6.3.1 Model Structure

Algorithm 1 proposed by Li et al. [2022] consists of three components: the generative module, the discriminative module, and the evaluation module. The generative module measures the distance between adjacent characters. Meanwhile, the discriminative module predicts explicit segmentation labels **B** and **I** for each character. **B** denotes the beginning of a word, while **I** denotes the inside of a word. In our task, we have tones instead of characters and segments instead of words. Therefore, one sentence corresponds to one chant.

Algorithm 1 Unsupervised Word Segmentation Procedure

```
Require: Generative Module $G$, Discriminative Module $D$, Evaluation Module
  $E$, sequences to be segmented $X$.
  $i = 0$
  while True do
      Segment the sequences $X$ with $G$ into $X^g$
      Transform the segmented $X^g$ into "BI" labels
      Train $D$ with high confident segmentations in $X^g$
      Segment the sequences $X$ with updated $D$ into $X^d$
      Train G with high confident segmentations in $X^d$
      Evaluate the segmented sequence $X^d$ with $E$ $e = E(X^d)$
      if $e^i < e^{i-1}$ then
          Return $D^{i-1}$
      end if
      $i+ = 1$
  end while
```

Within the generative module of the BERT model, the first step is to perform masked language modeling with a probability of $r = 0.12$ to mask each tone and obtain the so-called *masked_lm_loss*. Then, the model applies an additional linear layer and a softmax layer to predict the **B** or **I** labels. The softmax prediction is used to compute the score of the tone labels at position i:

$$\text{labels\_score}[i] = \text{softmax\_output}[i][0] - \text{softmax\_output}[i][1]. \tag{6.31}$$

If labels_score[i] is a positive number, it is likely that the tone should be labeled as **B**. Otherwise, a prediction with the **I** label is more probable. The next step is to train the model to learn the vector distance between tones based on the predicted labels_score. The input sequence of tones is copied into $2 \cdot n - 1$ copies, where n is the length of the chant, implying the chant consists of n tones. Let's split these copies into two sets. The first would consist of n copies, the second one of $n - 1$ copies. In the first set, in the i-th copy, the i-th tone is masked. In the second set, in the i-th copy, the i-th and $(i+1)$-th tones are masked. Then, the vectors of these sequences are calculated using the BERT model. The vector of the i-th tone in the i-th copy from the first set is denoted as H_i. So, when we only mask the i-th tone, we are interested in its vector. Furthermore, let's denote the vector of the i-th tone in the i-th copy from the second set as $H_{i,j}$, where $j = i+1$, and let's denote the vector of the $(i+1)$-th tone in the i-th copy from the second set as $H_{j,i}$. This is a case where we mask two adjacent tones, and we are interested in the vector from the first masked tone and then from the second masked tone. If the tone at position i in the chant and the tone at position $j = i+1$ should belong to the same segment, then the distance between the vectors H_i and $H_{i,j}$ should be large. However, if these tones are not strongly connected, then the vectors H_i and $H_{i,j}$ should be similar. The distance is therefore defined as:

$$d[i] = \frac{dist(H_i, H_{i,j}) + dist(H_j, H_{j,i})}{2}, \tag{6.32}$$

where $dist(a,b)$ represents the Euclidean distance. We consider the value d as the prediction and generate the targets by correcting the value of d. We iterate over each tone i in the chant, and if $d[i] > threshold_B$ and labels_score$[i] \geq 0.5$, the target value of the tone i is going to be $threshold_B$. This means that if BERT is confident about a tone being labeled as **B**, but the distance does not match, we want to retrain it. The same applies to **I**, so if $d[i] < threshold_I$ and labels_score$[i] \leq -0.5$, the target value is $threshold_I$. Otherwise, the target value remains $d[i]$. Then, the *generative_loss* is computed as the mean squared error loss between the targets and d. The final loss function used for backpropagation in generative training is a combination of these two losses:

$$loss = 0.8 \cdot masked\_lm\_loss + 0.2 \cdot generative\_loss. \tag{6.33}$$

The discriminative module computes the *masked_lm_loss* in the same way as the generative module. Then, the input sequence of chant tones is again copied $2 \cdot n - 1$ times, assuming n to be a length of chant (number of chant melody tones). We compute d for each tone using Equation 6.32. Then, for each tone i, target labels are created such that if $d[i] \geq threshold_I$, the target is 1, if $d[i] < threshold_B$, the target is 0. Otherwise, the target value is -100. The target 1 encodes the **I** label, while 0 encodes **B**. The label -100 is ignored during training. These targets are used to compute the *discriminative_loss* as the cross-entropy of an extra linear layer of BERT and its softmax output for predicting the **I** and **B** labels. The final loss of the discriminative module is then calculated and backpropagated as:

$$loss = 0.3 \cdot masked\_lm\_loss + 0.7 \cdot discriminative\_loss. \tag{6.34}$$

Initially, the model is trained with two epochs using the discriminative module. It is then iteratively trained for several epochs by iterating the generative and then the discriminative module. In each iteration, 3200 chants were used for training. The first 1600 chants are used to train the BERT model using the generative module. The second 1600 chants are used to train the model using the discriminative module.

6.3.2 Pretraining

The training process of Li et al. [2022] used the BERT model pretrained for the Chinese language provided by Wolf et al. [2019]. This is a crucial and powerful part of the training process of the BERT model that we cannot reproduce. Therefore, we have defined a new model with the same configuration as the Chinese model. We only create a new vocabulary corresponding to the Volpiano notation's tone characters. Then, we pretrain the model using masked language modeling with a masking probability of 0.15 and train it for 40 epochs.

7. Experiments

In this chapter, we will introduce our experiments and their results. First, we will introduce some baselines that could be useful for comparison. Then we will show the results of all score functions applied to our models and compare the score functions' results to the baselines. We also consider the best segmentation of antiphons and responsories to extract and analyze top 100 melodic units. As the final measurement, we will discuss the experiment of removing segments from different sides of the chants to observe the importance of the segments positioned at a specific part of the chants.

Unigram Model Settings

We explored four versions of the unigram model with different settings. The first one, which we referred to as *UM3_5*, is a unigram model that allows only segments of length between three and five tones. Another model, *UM1_7*, is a unigram model of segments whose allowed range is from one to seven tones. We also explored models with the mode extension, which states that each mode has its own model variables. The model first needs to predict the chant's mode to be able to do the segmentation with the most relevant variables. This way, the segmentation is not affected by mode gold data, so it is done in a proper way. The unigram model of the allowed segments in the range of three to five that is based on this extension is referred to as *UMM3_5*, and the one with a range of one to seven is referred to as *UMM1_7*. All these models are trained for 100 iterations with the default settings as it was described in Section 6.1.

NHPYLM Model Settings

This work measures two variants of NHPYLM. The first one is the basic NHPYLM model trained on 200 iterations with $L = 7$ maximal segment size, using default hyperparameters as was described in Section 6.2. This chapter refers to this model as *NHPYLM*. The second variant is the NHPYLM with the mode extension that we refer to as *NHPYLMModes* model. We trained this model on 20 iterations since it takes much computational time to validate hyperparameters on the validation dataset. The rest of the hyperparameters are also set the same way, as in the case of the basic NHPYLM model.

BERT Model Settings

The BERT model is first pretrained on 40 iterations optimizing the masked language modeling for each dataset separately. Then, the training process itself optimizes the pretrained model. In the case of both antiphon datasets, the training process consists of two initial epochs using the discriminative module and four epochs of generative-discriminative training. Overall, there were 11 iterations of the generative-discriminative process. Regarding both antiphon datasets, we set $threshold_B = 8.6$ and $threshold_I = 11.5$. In the case of the responsory dataset, there are also two initial epochs of the discriminative module and four epochs of the generative-discriminative loop. Because of less data in the responsory dataset, four epochs mean 5 iterations of the generative-discriminative process. In the context of responsories, thresholds are set to $threshold_B = 5.5$ and $threshold_I = 6.5$.

7.1 Baselines

In order to discuss the quality of the segmentations predicted by our models, we also have to measure the baselines of evaluation metrics using some very naive approaches that we should outperform. The most intuitive approach is randomly generated segmentation. We use the same segmentation as the one used as the initial segmentation in the unigram model as described in Section 6.1. So for each chant, we iteratively generate a random number from the Gaussian distribution $x \sim \mathcal{N}(\mu, \sigma^2)$, for $\mu = 5$ and $\sigma = 2$, and then we take the $l = min(max(1, n), 7)$ as the length of the next segment, if possible. We refer to this baseline segmentation as *Rand*.

Cornelissen et al. [2020b] already proposed several segmentations. They concluded that the natural segmentations were the most promising ones. By natural segmentation, we mean the segmentation generated by the chant text. According to Cornelissen et al. [2020b], antiphons segmented by words showed the best results. Meanwhile, syllable segmentation was the best for the responsory set of chants. We consider these segmentations for comparison with our proposals on the mode classification task as well as on other evaluation metrics we proposed. Ideally, we would want to defeat those segmentations. Otherwise, we should show that there is no better segmentation. To be able to compare their proposals with ours, we have to reevaluate the specific segmentation with the same settings as they did on all our evaluation functions. For instance, we have to keep liquescents. Cornelissen et al. [2020b] measured the experiment five times with different seeds of splitting the train and test datasets. We can not do that for each model due to the high computational time. So we evaluate these segmentations only once and use the same dataset split for all models and baselines as described in Section 4.1. We denote the original segmentation by words proposed by Cornelissen et al. [2020b] as *Words_liq*, and the segmentation by syllables presented by them is called *Syllables_liq*. We also measured these two segmentations after replacing liquescents with their basic tone representations. We call these segmentations *Words* and *Syllables*. We also replace liquescents in the same way for the Rand baseline, the upper bound we will describe below in this section, or any of the models we proposed. Since Cornelissen et al. [2020b] misused the antiphon dataset (they also included differentiae - parts of succeeding, eventually preceding psalms - as part of the antiphon chants), we need to reevaluate other approaches proposed by them on the third dataset, the antiphon-without-differentiae dataset, to confirm their statements and results. One of these approaches was n-gram segmentation, meaning each segment has a fixed length of n as long as possible, starting at the beginning of the chant. The last segment of each chant adapts to the chant melody and has a smaller or equal length than n. We take into consideration the 4-gram segmentation as part of the antiphons-without-differentiae dataset results, and we call it *4gram_liq*. This segmentation also keeps liquescents in order to be able to compare its results with the antiphon experiments of Cornelissen et al. [2020b].

We already mentioned segmentations that we should or would want to defeat.

But we still need to figure out the upper boundaries of our evaluation metrics that we cannot outperform for sure. Even if there is no good or better segmentation than the one based on the chant text, it is essential to provide some evidence. In the case of most evaluation functions we defined in Chapter 5, it is not straightforward what is the ideal score value - if its value is better to be lower or higher. On the other hand, in the mode classification task, we know that the higher f1 and accuracy, the better. Therefore, we propose the mode classification approach, considering all overlapping n-grams as features for $n \in \{1, ..., 7\}$. In the case of bacor_score or nb_score, we remove the TF-IDF vectorizer limit on the features, which is by default set to be 5000. Therefore, the classification model has all possible segments of length $1, ..., 7$ (e.g., all possible segmentations) of each chant available. It is not segmentation, so we can not use that as segmentation to solve our task. On the other hand, we can use this approach to be the upper bound of the mode classification task using segmentations. We refer to this approach as *NgramOverlap*. The only disadvantage is that the model also has a lot of misleading segments. Furthermore, we also measured the same approach for fixed n. Therefore, we include results of the *6gramOverlap*, the set of all possible segments of length 6 of each chant, as part of the results of the responsory dataset since the results seem interesting.

7.2 Results

In this section, we will discuss the results of the score functions measuring baselines and segmentations generated by our proposed models. First, we will look at the three dataset's score values separately. Then, we will discuss the results of the mode classification tasks in detail. And finally, we will look at the properties of the best segmentations.

Antiphon Dataset

We measured the scores for this dataset to compare our results with Cornelissen et al. [2020b]. As we mentioned, the dataset chants contain the differentiae that are not part of antiphons but part of the psalm corresponding to the particular antiphon. Furthermore, there is only a limited set of differentiae, while that set is strongly related to the modes. Table 7.1 shows the scores related to the mode classification tasks, which is the one that Cornelissen et al. [2020b] was dealing with. Table 7.2 contains the model perplexity and fundamental segmentation analysis such as vocabulary size, average segment length, and vocab_levenshtein that were described in Section 5.2. The rest of the value scores, such as wtmf, maww, mawp and wufpc are shown in Table 7.3.

Regarding the mode classification task based on segmentation, the Rand model, compared with other approaches or baselines, says that looking for a good segmentation makes sense. The baseline of Words is quite strong, and retaining liquescents is detrimental. As Table 7.1 shows, we can reach better performance similar to the tentative upper bound of NgramOverlap. Moreover, we can get good results even using the score functions based on the Naive Bayes classifier when we use the mode extension of our proposed models.

	bacor_accuracy	*bacor_f1*	*nb_accuracy*	*nb_f1*
Rand	87.59	87.39	81.37	82.20
Words_liq	94.76	94.71	90.17	90.30
Words	95.22	95.18	91.01	91.10
UM3_5	93.58	93.53	88.15	88.43
UM1_7	94.52	94.47	88.70	89.06
UMM3_5	93.99	93.98	93.03	93.03
UMM1_7	92.69	92.64	83.34	83.01
NHPYLM	95.77	95.75	93.94	94.03
NHPYLMModes	**96.03**	**96.03**	**96.18**	**96.18**
BERT	89.52	89.39	81.80	82.26
NgramOverlap	*96.13*	*96.11*	*92.74*	*92.59*

Table 7.1: Antiphon dataset - mode classification scores.

	perplexity	*vocab_size*	*avg_seg_len*	*vocab_levenshtein*
Rand	-	14986	4.28	0.91
Words_liq	-	11087	3.77	0.91
Words	-	9434	3.77	0.90
UM3_5	818.79	2325	4.26	**0.99**
UM1_7	1517.45	3581	4.96	0.93
UMM3_5	441.70	3143	4.06	0.97
UMM1_7	367.89	3782	3.85	0.92
NHPYLM	**25.20**	**2161**	2.44	0.90
NHPYLMModes	28.10	3493	2.78	0.93
BERT	-	6300	1.85	0.82

Table 7.2: Antiphon dataset - perplexity, vocabulary size, average segment length, and vocab_levensthein scores.

The perplexity score indicates the statistical quality of the NHPYLM model. The NHPYLMModes model is slightly less confident, but the score value is still very promising. The difference of perplexity between those two models could be caused by the fact that due to the computational time, the NHPYLM model was trained on the 10 times more training iterations. Eventually, the difference could be caused by the different random sampling during the training. However, the perplexity difference is small enough to state that it is not certain which model is better. Regarding the unigram model, the perplexity of the UM3_5 is expected to be lower than the perplexity of the UM1_7 model since the UM3_5 model has less possible segments that it works with and considers during training and predicting processes. Therefore, it is interesting that the UMM1_7 model has lower perplexity than the UMM3_5 model, so the UMM1_7 seems to be trained better.

Compared with the baselines of both Rand and Words, we can see that our proposed models reduce the vocabulary size during training. Hence, the segmentation is better suited to memorization. The vocab_size value alone suggests that segmentation by words is not the perfect approach. The smallest vocab_size is provided again by the NHPYLM model. Even though our models with the mode extension have eight different vocabularies and different variables for each mode, and chants of each mode are trained separately, the vocab_size of UMMs and NHPYLMModes is not eight times bigger than the vocab_size of the basic model variant. It is not twice as big. But the vocabulary is still larger. This means that each mode can have specific characteristic melodic units, just as there are melodic units common to all modes.

The score vocab_levenshtein provides information about the divergence of melodic units in the vocabulary, as described in Chapter 5. The higher score, the more divergent the vocabulary is. The maximal value of vocab_levenshtein is 1.00. Most models provide a score greater than 0.90 which is also approximately the baseline value. The UM3_5 model's segmentation is trained to be divergent the most with the vocab_lveneshtein to be 0.99, where similar melodic units tend to be grouped to only one segment representation. On the other hand, the value of this score function of the BERT model is 0.84. This neural network approach groups tones that behave similarly regarding the chant melody context. Therefore, it could easily happen that two different melodic units contain the same subsegment, which is, for instance, shifted and grouped with more or less different tones. In this case, using the common part as a separate melodic unit would be better. The BERT's vocabulary divergence is not optimal, and the vocabulary size is almost three times larger than the vocabulary size of the NHPYLM, despite the lower avg_seg_len. Therefore, this approach is not suitable for this task. Note that, in general, the neural network approaches need a large amount of training data that we can not provide, which could be the main reason for BERT's weak performance on our task.

	wtmf	*maww*	*mawp*	*wufpc*
Rand	64.61	26.89	37.20	5.82
Words_liq	63.00	100.00	-	6.23
Words	61.71	100.00	-	5.73
UM3_5	54.91	32.76	46.83	5.59
UM1_7	57.92	33.10	51.63	5.33
UMM3_5	64.21	34.45	48.23	5.71
UMM1_7	59.90	38.66	56.49	5.85
NHPYLM	49.27	47.87	71.81	7.73
NHPYLMModes	60.50	49.32	67.23	6.27
BERT	48.09	57.75	64.38	6.92

Table 7.3: Antiphon dataset - weighted top mode frequency, melody aligned with words, melody aligned with phrases, weighted unique final pitch count scores.

The wtmf score gives the knowledge about how frequently melodic units of the vocabulary occur in its dominant mode. The dominant mode is the one that has the most occurrences of the particular melodic unit. As we can see from Table 7.3, it is not a surprise that models based on the mode extension provide higher wtmf than the model's basic variant. In the case of the mode extension-based variant, we force the model to treat each mode separately. On the other hand, all models and baselines provide a high wtmf score, even the Rand baseline. If all melodic segments are uniformly distributed over all modes, then the wtmf would be 12.5. Therefore, most of the melodic units of the vocabulary are used primarily in chants of its dominant mode, but they also occur, not that frequently, in chants of other modes. This is a significant result that demonstrates the existence of the relation between modality and segmentation. As we can see, the wtmf score of the Rand method is the highest one, which could be explained by the largest vocabulary with the most unique and least frequent segments. Note that the wtmf score is the weighted average over all vocabulary segments, where more frequent segments impact the value more significantly. We can see that models based on the mode extension provide similar outcomes, even though each mode is trained separately without any knowledge of chants belonging to other modes.

Suppose we compare the Rand baseline with proposed models in Table 7.3. We can notice that segmentations generated by models with better perplexity or with better mode classification performance are more aligned with words and phrases (maww, respectively mawp scores). Therefore, the segmentation quality seems to relate to the word and phrase boundaries. But similarly, on average, segmentations with shorter segments tend to be aligned with words and phrases better. On the other hand, approximately 30 % of words do not end at the same tone as any of the segmentations segments that perform better on the mode classification task than the segmentation by words. This is true even though those better segmentations have avg_seg_len lower than the Words baseline, so those segments should be aligned with words more easily. Therefore, this is another indication that the segmentation by words as proposed Cornelissen et al. [2020b] is not that strong.

The last score we have not discussed is the wufpc score. As we can see, the value of the weighted unique final pitch count is similar to all baselines and model segmentations. The difference seems to be caused by the average segment length value kept in the avg_seg_len score. The lower the avg_seg_len, the more segments are predicted in the chants and the more final segment pitches the chants have. Therefore, this score is useless for now in measuring the segmentation quality.

Antiphons-Without-Differentiae Dataset

In the previous section, we discussed the results of the antiphon dataset containing differentiae. This dataset was used by Cornelissen et al. [2020b] for their experiments. Based on those experiments, they concluded that natural segmentation is the strongest one. However, their results from the antiphon dataset are irrelevant to the mode classification task based on chant melodies because they used information indicating the antiphonal context of psalms and their modes. In this section, we are going to discuss reevaluated results of the corrected antiphon dataset of Words_liq and 4gram_liq that Cornelissen et al. [2020b] proposed and concluded that segmentation by words is better than the 4-gram segmentation. We compare these baseline segmentations and their scores of our scoring function with our proposed approaches in Tables 7.4, 7.5 and 7.6.

Table 7.4 shows that the conclusion that "natural" segmentation is better because it leads to better mode classification results by Cornelissen et al. [2020b] based on their approaches is wrong since the 4gram_liq has a higher score of the mode classification task. Compared with Table 7.1, the Words_liq's score decreased by almost 5 %. This means that differentiae segmented by words are strong, but not the segmentation itself. The performance of the unigram models on the mode classification task again is not that good. Meanwhile, the NHPYLM model again performs the best, especially the one with the mode extension. But

	bacor_accuracy	*bacor_f1*	*nb_accuracy*	*nb_f1*
Rand	81.70	81.08	75.33	76.68
Words_liq	90.28	90.16	86.57	86.84
Words	90.53	90.38	86.72	86.98
4gram_liq	91.27	91.14	83.25	83.62
UM3_5	90.11	90.00	84.50	84.94
UM1_7	89.84	89.69	84.99	85.50
UMM3_5	89.82	89.77	87.43	87.40
UMM1_7	88.12	87.91	76.21	75.92
NHPYLM	92.99	92.90	91.07	91.31
NHPYLMModes	**94.02**	**94.01**	**93.58**	**93.59**
BERT	87.28	87.11	79.46	80.02
NgramOverlap	*94.69*	*94.65*	*90.14*	*89.95*

Table 7.4: Antiphons-without-differentiae dataset - mode classification scores.

the upper bound model NgramOverlap indicates that there still could be an improvement in the segmentation prediction. Section 7.2.1 will discuss our results of the mode classification task based on segmentation in more detail.

	perplexity	*vocab_size*	*avg_seg_len*	*vocab_levenshtein*
Rand	-	14281	4.26	0.91
Words_liq	-	10780	3.63	0.91
Words	-	9201	3.63	0.90
4gram_liq	-	4211	3.89	1.00
UM3_5	825.84	**2270**	4.25	**0.99**
UM1_7	1511.34	3407	4.83	0.93
UMM3_5	525.07	3122	4.01	0.98
UMM1_7	401.23	3731	3.58	0.92
NHPYLM	**26.10**	2353	2.34	0.90
NHPYLMModes	31.08	3317	2.69	0.93
BERT	-	6654	1.97	0.84

Table 7.5: Antiphons-without-differentiae dataset - perplexity, vocabulary size, average segment length, and vocab_levensthein scores.

Results in Table 7.5 are similar to Table 7.2 as well as the Table 7.6 is similar to Table 7.3. The behavior of baselines and models are similar on both antiphon datasets. In the previous section, we discussed their properties and their relations to evaluation functions in detail. The antiphons-without-differentiae dataset appears to be a bit complex and not that straightforward to be trained on. Models are again able to decrease the vocabulary size. The vocabulary seems divergent enough except for the BERT model, which is unsuitable even for this dataset. On the other hand, the NHPYLM also has the most promising results on the segmentation of the antiphons-without-differentiae dataset, especially the model variant based on the mode extension. Baseline segmentations and those generated by the proposed models indicate that each mode has some characteristic melodic units used primarily in the specific mode. Still, many segments are shared among all modes. Results discussed till now already suggested that word segmentation is

	wtmf	*maww*	*mawp*	*wufpc*
Rand	63.39	27.37	37.44	5.79
Words_liq	61.50	100.00	-	6.22
Words	60.09	100.00	-	5.72
4gram_liq	55.05	29.51	-	6.36
UM3_5	52.79	32.39	48.79	5.58
UM1_7	52.44	31.07	52.44	5.38
UMM3_5	60.75	34.29	51.26	5.61
UMM1_7	54.91	37.74	58.13	6.00
NHPYLM	49.63	46.78	71.59	7.72
NHPYLMModes	55.99	47.78	67.91	6.28
BERT	47.71	52.74	62.84	7.19

Table 7.6: Antiphons-without-differentiae dataset - weighted top mode frequency, melody aligned with words, melody aligned with phrases, weighted unique final pitch count scores.

not probably the best approach, which is supported by the maww score measured on this dataset, where approximately 70 % of words do not end at the tone where ends any of 4gram_liq segments. On the other hand, the same as considering the previous incorrect dataset, the maww increase on segmentation with better performance on perplexity and mode classification scores. The mawp also seems related to the segmentation quality. Note that it is still not clear whether the best segmentation should be aligned with 100 % of the phrases.

Responsory Dataset

The third dataset consists of longer and more complex chants than antiphons. Responsories do not contain differentiae, so we do not need to remove them. Therefore, we can directly follow up on the Cornelissen et al. [2020b]. In this case, they concluded that the segmentation by syllables is the most promising one. So we will take that as a baseline, which we also do with the syllable segmentation after removing liquescents, and we will compare these results with our methods. Results of all value evaluation scores are listed in Tables 7.7, 7.8, 7.9. The results of baselines and models are similar to those evaluated on both antiphon datasets.

As in the previous evaluation of antiphon datasets, in the context of the mode classification, the unigram model cannot beat the natural segmentation (in this case, the one based on syllables). On the other hand, NHPYLM and NHPYLM-Modes again give better performance. Furthermore, considering the overlapping n-grams of fixed $n = 6$ 6gramOverlap, we get better bacor_accuracy and bacor_f1 score than the NgramOverlap which consists not only the same features as 6gramOverlap does but also all other overlapping n-grams for $n \in \{1, ..., 7\}$. That could indicate that responsories contain powerful and important segments of six tones. However, the nb_accuracy and nb_f1 scores of the 6gramOverlap are significantly lower than those scores of NgramOverlap. Therefore, if there are such strong segments, selecting them for the mode prediction is probably not straightforward. We will discuss the detailed results of the mode classification task in Section 7.2.1.

	bacor_accuracy	*bacor_f1*	*nb_accuracy*	*nb_f1*
Rand	82.12	81.93	75.11	76.09
Syllables_liq	92.70	92.68	89.81	89.95
Syllables	93.27	93.25	89.43	89.55
UM3_5	92.18	92.13	84.73	84.89
UM1_7	92.41	92.38	86.39	86.59
UMM3_5	91.18	91.18	90.61	90.62
UMM1_7	89.47	89.45	79.66	78.94
NHPYLM	93.12	93.12	91.13	91.23
NHPYLMModes	**94.22**	**94.22**	**94.22**	**94.21**
BERT	87.43	87.37	75.91	76.49
NgramOverlap	*94.31*	*94.30*	*93.22*	*93.20*
6gramOverlap	*95.21*	*95.20*	*91.99*	*91.92*

Table 7.7: Responsory dataset - mode classification scores.

	perplexity	*vocab_size*	*avg_seg_len*	*vocab_levenshtein*
Rand	-	16839	4.37	0.91
Syllables_liq	-	7342	2.92	0.90
Syllables	-	6907	2.92	0.90
UM3_5	978.36	**2625**	4.44	**0.99**
UM1_7	1972.99	4443	5.22	0.94
UMM3_5	523.54	3336	4.16	0.98
UMM1_7	475.59	4447	4.11	0.94
NHPYLM	**22.92**	2676	2.68	0.92
NHPYLMModes	24.99	4170	2.93	0.93
BERT	-	4862	1.42	0.76

Table 7.8: Responsory dataset - perplexity, vocabulary size, average segment length, and vocab_levensthein scores.

	wtmf	*maww*	*mawp*	*wufpc*
Rand	57.59	26.22	27.20	7.11
Syllables_liq	49.75	100.00	-	9.22
Syllables	49.37	100.00	-	7.63
UM3_5	47.49	35.31	39.37	7.03
UM1_7	52.94	36.29	44.60	6.89
UMM3_5	56.93	38.23	43.27	7.05
UMM1_7	56.29	41.30	54.67	7.23
NHPYLM	46.15	55.49	76.52	8.84
NHPYLMModes	53.96	54.06	68.80	7.53
BERT	41.04	69.23	81.04	8.43

Table 7.9: Responsory dataset - weighted top mode frequency, melody aligned with words, melody aligned with phrases, weighted unique final pitch count scores.

Tables 7.1 to 7.9 show that our proposed models work on both chant types similarly. Scores are even more similar, comparing the responsory with the antiphon-without-differentiae dataset instead of the antiphon dataset used by Cornelissen et al. [2020b]. Again all models learn to decrease the number of melodic units in the vocabulary. At the same time, scores such as perplexity, maww, mawp, avg_seg_len, and those based on the mode classification correlate together. For the same reason as in the discussion of the antiphon dataset, responsorial segments are primarily used in one particular mode but can also be shared with other modes. Furthermore, regarding values of the vocab_levenshtein, avg_seg_len, and mode classification tasks, BERT is unsuitable even for the responsory dataset.

7.2.1 Natural Segmentation Is Not Ideal

Considering the antiphon chants, differentiae segmented in the proper way strongly affect mode classification results. Differentiae are mapped to the separated text that does not relate to the text of the rest of the melody. Therefore, when we segment by words, we for sure find the differentia melodic units. These melodic units have a strong modal identity, which caused that Cornelissen et al. [2020b] measured word segmentation of antiphons as the approach with the best mode classification score. Replacing liquescents with their original tone representations, we would get an even better score. However, we proposed a method that, on the antiphon dataset, has a better mode classification score. This indicates that Cornelissen et al. [2020b] conclusion of segmentation by natural units being the best is not correct. Furthermore, if we remove differentiae, which should not be considered as part of antiphon melodies, the mode classification score of segmentation by words is decreased by almost 5 %. Using the antiphons-without-differentiae dataset, the 4gram_liq segmentation, also proposed by Cornelissen et al. [2020b], outperforms the segmentation by words.

Considering the antiphon dataset, the same as the Cornelissen et al. [2020b] used, we provide the state-of-the-art of mode classification task. On the other hand, this dataset is not relevant for us anymore since differentiae strongly affect the dataset. Therefore, for the rest of the work, we consider only the results of the antiphons-without-differentiae and responsory datasets, where we provide the state-of-the-art on the mode classification task as well. In the case of the antiphons-without-differentiae, the best approach so far is overlapping n-grams of $n \in \{1, .., 7\}$ collected together with the accuracy of 94.69 % and f1 score 94.65 %. Meanwhile, in the case of responsories, overlapping 6-grams offered the best results with an accuracy of 95.21 % and f1 score of 95.20 %.

The goal of this work is not to find the optimal approach to the mode classification task. As discussed in Section 5.3.1, we hypothesize that correctly predicting a chant mode based on melody segmentation is one possible measure of segmentation quality. The assumptions seem to be correct because there are other evaluation functions, such as perplexity or mawp, whose improvement correlates with an improvement in the mode classification scores, as could be seen in Figure 7.1. Furthermore, it is not straightforward what is the potential of segments as features to classify modes. As discussed in the baseline section, we use the NgramOverlap as the upper bound of the mode classification task using chant's segments as features. That is a good indicator that there is a potential to find better segmentation than baselines Rand and those proposed by Cornelissen et al. [2020b] (Words_liq, Syllables_liq or 4gram_liq). On the other hand, the NgramOverlap approach is not segmentation, and it is not certain that any segmentation could reach a similar accuracy and f1 score on mode classification. However, segmentations generated by NHPYLM models, especially the NHPYLMModes model, are already very close to these upper bound values.

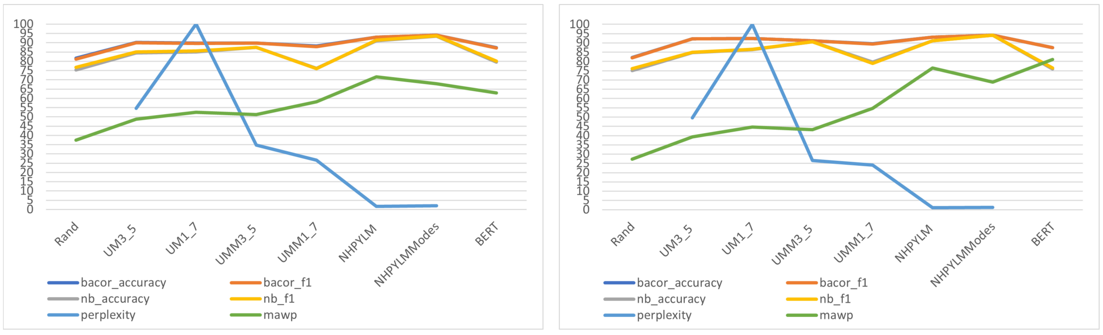


Figure 7.1: Charts visualizing correlation between bacor_accuracy, bacor_f1, nb_accuracy, nb_f1, perplexity and mawp. Perplexity values are normalized to be in the range of 0 and 100, so each perplexity value is divided by the highest one, and the result is multiplied by 100. Scores of antiphons-without-differentiae dataset are shown on the left. The right chart visualizes the correlation of responsory dataset scores.

7.2.2 The Importance of the Beginning, Middle, and End of the Chant

Regarding and analyzing Tables 7.1 to 7.9, the most promising segmentations of Gregorian chants are provided by the NHPYLMModes model and eventually the NHPYLM model. They have the best score on mode classification, the best perplexity, the highest mawp score, and other score functions that analyze these segmentations positively. In this section, we will look at the segment properties of segmentations provided by these two models considering both relevant datasets, antiphons-without-differentiae, and responsories.

The segmentation of both NHPYLM and NHPYLMModes has an average segment length of 2.34, and 2.69, respectively, in the case of the antiphons-without-differentiae. Regarding the responsory datasets, the avg_seg_len value of NHPYLM is 2.68, and the NHPYLMModes's avg_seg_len is evaluated to be 2.93. If we compare these numbers to avg_seg_len of other approaches, the values of both NHPYLM variants are lower. That could be caused simply by the shorter segments that were discovered. But it could also be caused by many single-tone segments not associated with neighboring tones. In the second case, the segmentation would contain a lot of single tones whose frequencies could also significantly impact the TF-IDF vectorizer that gives vector features we use as direct inputs for SVC or Naive Bayes classifiers. The NgramOverlap approach has the same problem when considering that there are also included all segments of length 1. That could be theoretically the reason for the improvement of bacor_accuracy and bacor_f1 scores using the NHPYLM variants or evaluating the NgramOverlap comparing with the rest of models and baselines. Therefore, we did the experiment again with the binary settings of TF-IDF, so term frequency is either 0 (the chant does not contain the segment) or 1 (the segment is included at least once in the segmented chant). But there were no significant differences in results, so the count of segments, especially the count of single-tone segments, does not particularly affect the performance. To observe the distribution of segment lengths over chants, we consider the chart of average segment lengths described in Section 5.3.3. Figures 7.2 and 7.3 show the average segment length distribution over all chants of the particular mode of segmentations generated by the NHPYLM and NHPYLMModes models respectively of the antiphons-without-differentiae dataset. Then, Figures 7.4 and 7.5 display the average segment length charts of segmentations provided by the same models, the NHPYLM and the NHPYLM-Modes models respectively, this time on the responsory dataset.

Remember that longer segments contribute to more bins in the chart than short ones. As we can see, it is not that common that the average segmentation length is as low as the avg_seg_len values. That could be caused by single-tones or generally small segments that lay among segments with a larger length that wasn't grouped together since the compounded melodic unit is not as typical in data as those separated parts. That may be due to the memorization mistakes mentioned in Chapter 2. However, it could also be caused by an inability to find segmentation because none would exist. In the second case, the predicted strong segments would be familiar melodic units that singers had easy to memorize, and they accidentally customized different melodies over the years to include these simple and common melodic segments (similarly as described Treitler [1975]). However, charts show us that grouping tones into segments at the end of chants is easier. In the case of the antiphon-without-differentiae dataset, also segments at the beginning tend to be longer, especially when considering the NHPYLMModes model.

Modality and the Sharing of Segments

At the beginning of Section 7.2, while commenting on the tables, we mentioned several reasons why we believe there are melodic units that are shared among multiple modes, but at the same time, there are melodic units that are predominantly used in chants belonging to a single mode. Are there also melodic units that are used only by one mode? We call these melodic units unique segments. In order to analyze the relation of modes and their segments, we look at the table of numbers of shared and distinct segments of each mode pair. Furthermore, we explore the vocabulary sizes of all modes separately and look at their number of vocabulary unique segments. Figure 7.6 shows the results of the antiphons-without-differentiae dataset, while Figure 7.7 analyzes the responsory dataset.
The number of shared and distinct segments between two modes correlates with the mode distribution over the dataset. Furthermore, all four pairs of related authentic and plagal modes (Section 1.2) also share more segments than most other pairs. The behavior of the segmentations examined in terms of shared segments, distinct segments, and vocabulary size is similar in both the NHPYLM and NH-PYLMModes models. On the other hand, the matrix of unique segment numbers is different. The NHPYLMModes model generates more unique segments for the particular mode that do not occur in any other modes. The significant difference is mainly in the modes whose records do not contain many chants, unlike the 1 or 8 modes. However, both models NHPYLM and NHPYLMModes generate

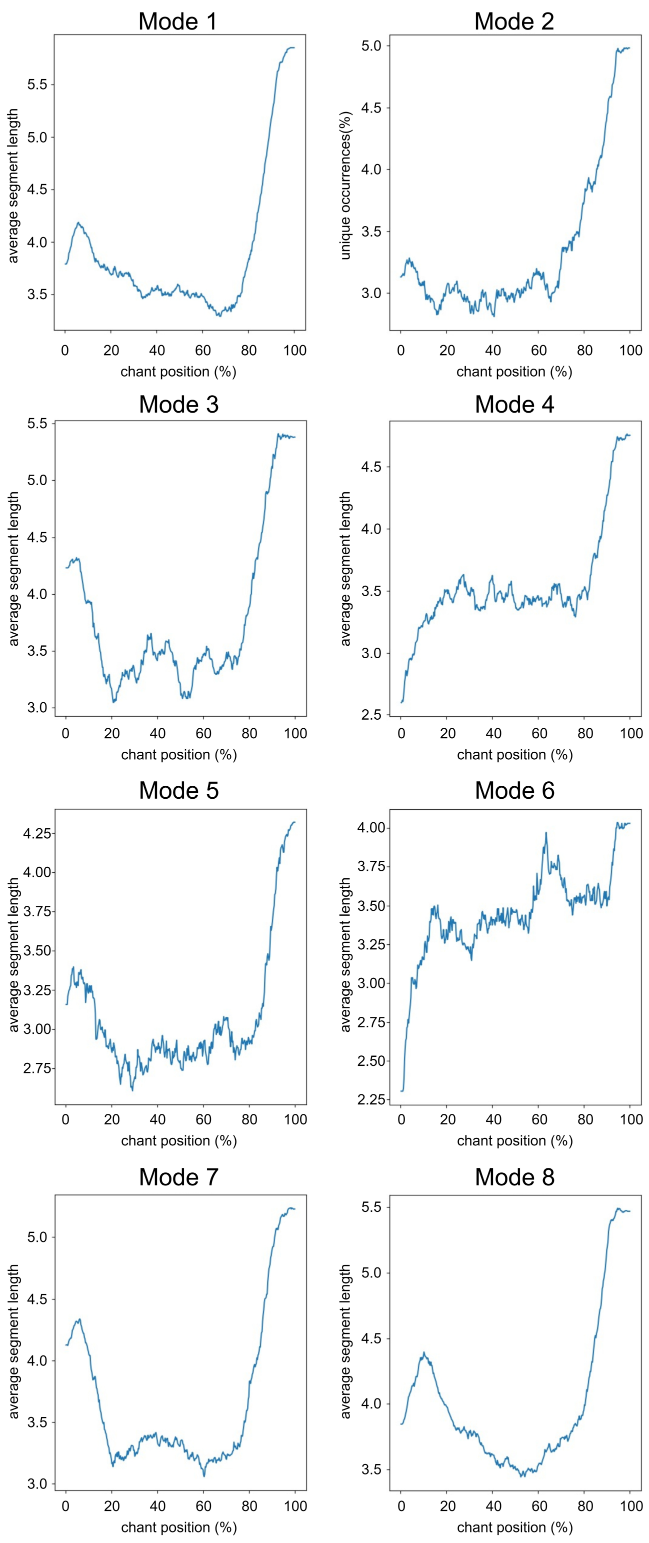


Figure 7.2: Antiphons-without-differentiae dataset - average segment length distribution charts for each mode separately of chant segmentation provided by NH-PYLM model. Each mode behaves differently, which could be caused by different mode properties as well as by not optimally trained the model regarding some of the modes. Usually, however, beginnings and primarily ends are, on average, longer than segments in the middle.

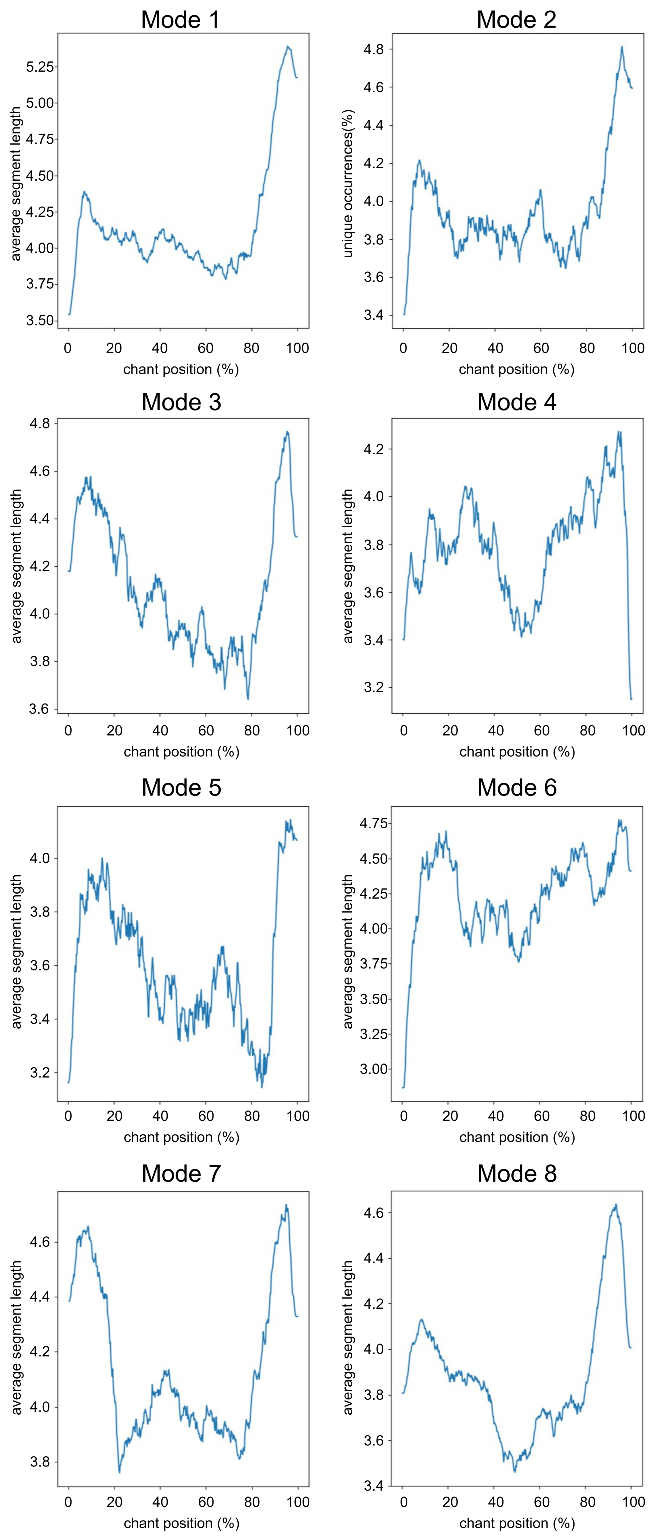


Figure 7.3: Antiphons-without-differentiae dataset - average segment length distribution charts for each mode separately of chant segmentation provided by NHPYLMModes model. All modes behave similarly than in the case of the NHPYLM model, which could indicate that the NHPYLMModes model is trained better.

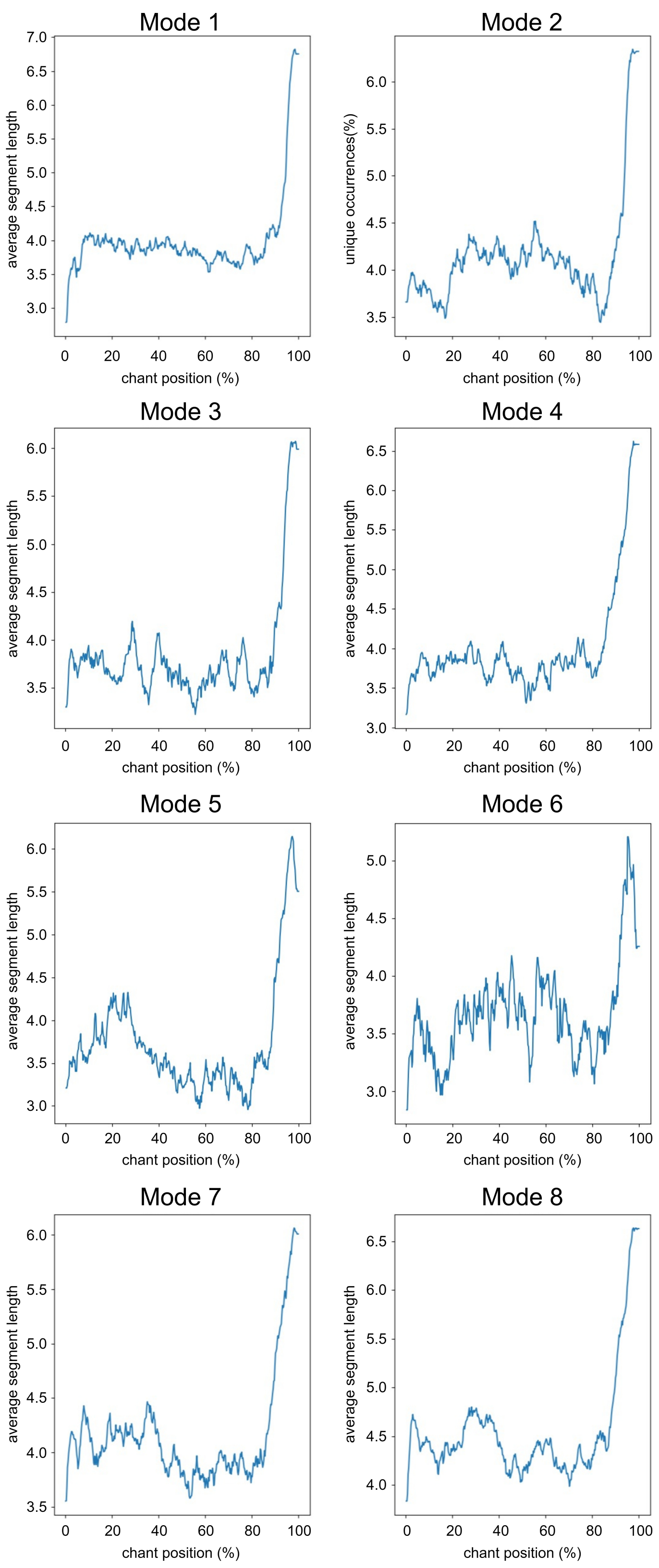


Figure 7.4: Responsory dataset - average segment length distribution charts for each mode separately of chant segmentation provided by NHPYLM model. In responsories, only the final segments are long. Unlike the antiphons, the beginning of chants is not that interesting. On the other hand, the final segments of responsories are, on average, longer than the final segments of antiphons.

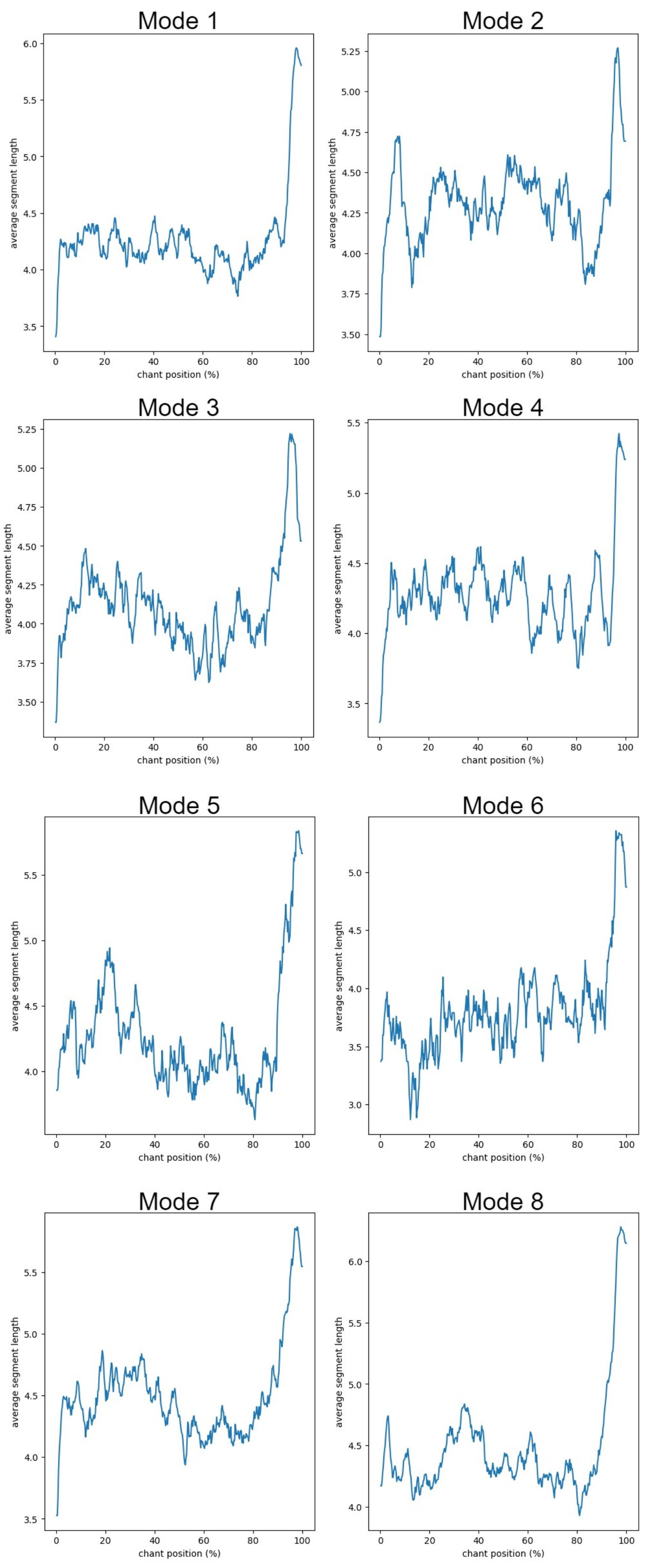


Figure 7.5: Responsory dataset - average segment length distribution charts for each mode separately of chant segmentation provided by NHPYLMModes model. The model finds longer segments at the beginning and in the middle of chants than in the case of the NHPYLM model.

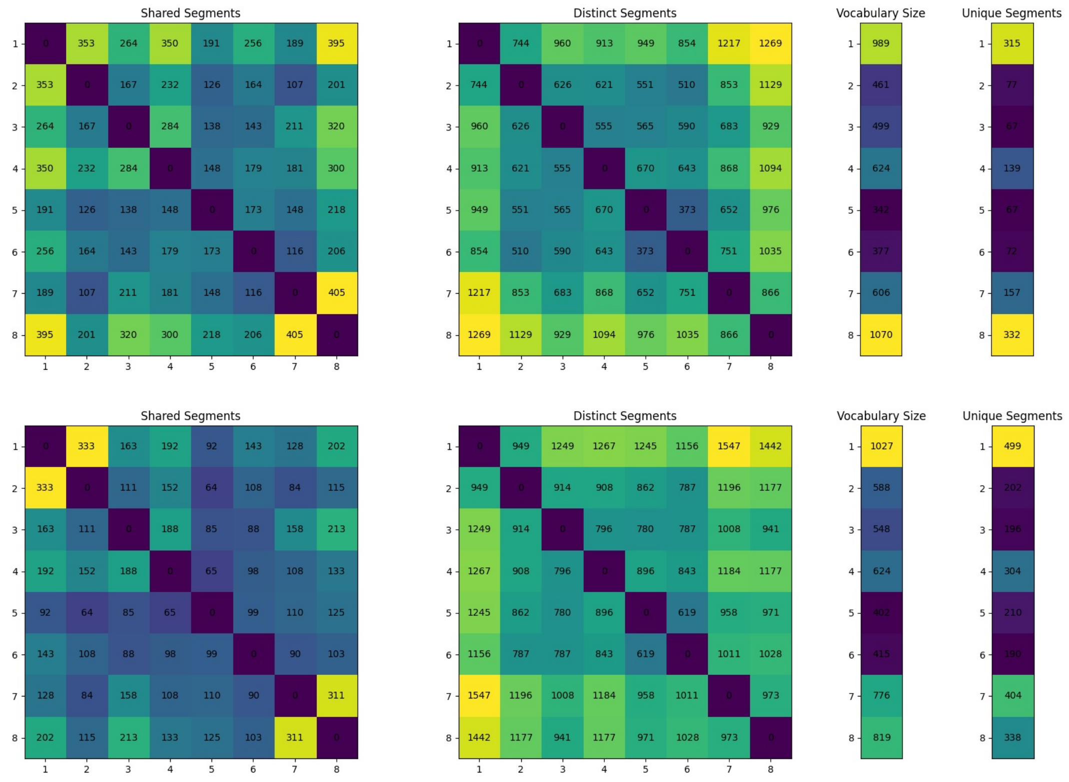


Figure 7.6: Antiphons-without-differentiae dataset - matrix of shared segments between two modes, matrix of distinct segments between two modes, sizes of vocabularies of segments occurred in chants of the specific mode, and the number of unique segments of the particular mode. The top matrices analyze the segmentation generated by the NHPYLM model. The bottom matrices analyze the segmentation generated by the NHPYLMModes model. In shared segments and distinct segments matrices, we cannot measure that value between the two same modes, so the values on the diagonal are filled to be zero.

approximately $10-50$ % of mode vocabulary segments to be unique segments. In Figures 7.8 and 7.9, we show the density of unique segments (Section 5.3.3) over the chant of each mode separately of the antiphons-without-differentiae dataset. Figures 7.10 and 7.11 display the same for the responsory dataset. Charts of both datasets indicate that unique segments are placed mainly at the beginning of chants. Antiphons seem to have also one more peak of unique segments occurrences in the second half of the chant. On the other hand, each mode has a slightly different curve. That could relate to different mode structures or indicate that the model is not trained enough for some modes. The responsory dataset is a little bit more complex. But in both datasets, the NHPYLMModes model provides more confident peaks and curves than the NHPYLM model.

We already measured the sizes of vocabularies. However, we still do not know anything about the frequency of the segments. Some melodic units could be used more often than others, or all melodic units could be used approximately uniformly. We analyze it using the last chart described in Section 5.3.3, the segmetns occurrences chart. Furthermore, these charts provide a lookup into

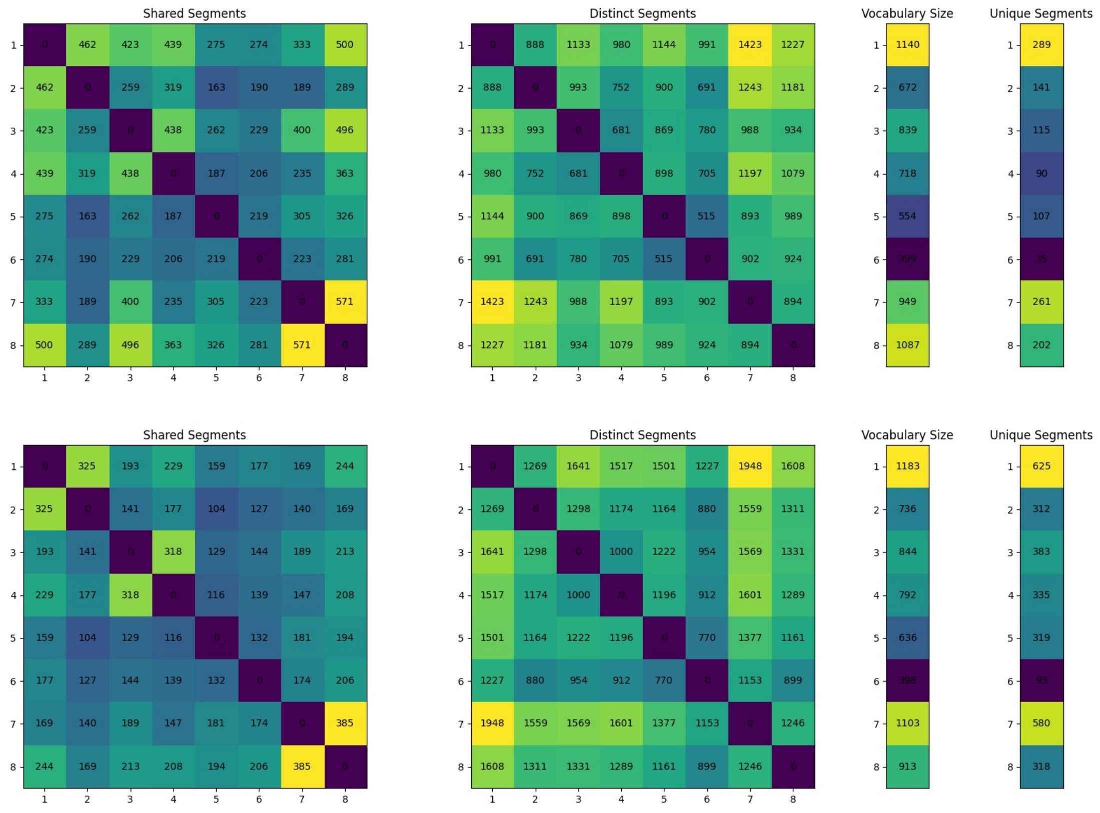


Figure 7.7: Responsory dataset - matrix of shared segments between two modes, matrix of distinct segments between two modes, sizes of vocabularies of segments occurred in chants of the specific mode, and the number of unique segments of the particular mode. The top matrices analyze the segmentation generated by the NHPYLM model. The bottom matrices analyze the segmentation generated by the NHPYLMModes model. In shared segments and distinct segments matrices, we cannot measure that value between the two same modes, so the values on the diagonal are filled to be zero.

wtmf score. More precisely, it shows how other modes use melodic units belonging mainly to one given mode. Figures 7.12 and 7.13 provide charts of segments occurrences for each mode as it was described in Section 5.3.3, where these two chart sets are measured on the antiphons-without-differentiae dataset using the segmentation by NHPYLM, and NHPYLMModes respectively. Figures 7.14 and 7.15 give the same charts for the responsory dataset.

The first observation is that authentic modes with the same final tone frequently share strong melodic units with their plagal variants (mode 1 with mode 2, mode 3 with mode 4, etc.). On the other hand, many strong melodic units are used primarily by its dominant mode. We can also notice a few powerful melodic units of the vocabulary that are used very often. The rest of the melodic segments are rarely used. So there are far fewer relevant melodic units than those found in the vocabulary using the NHPYLM or NHPYLMModes models.

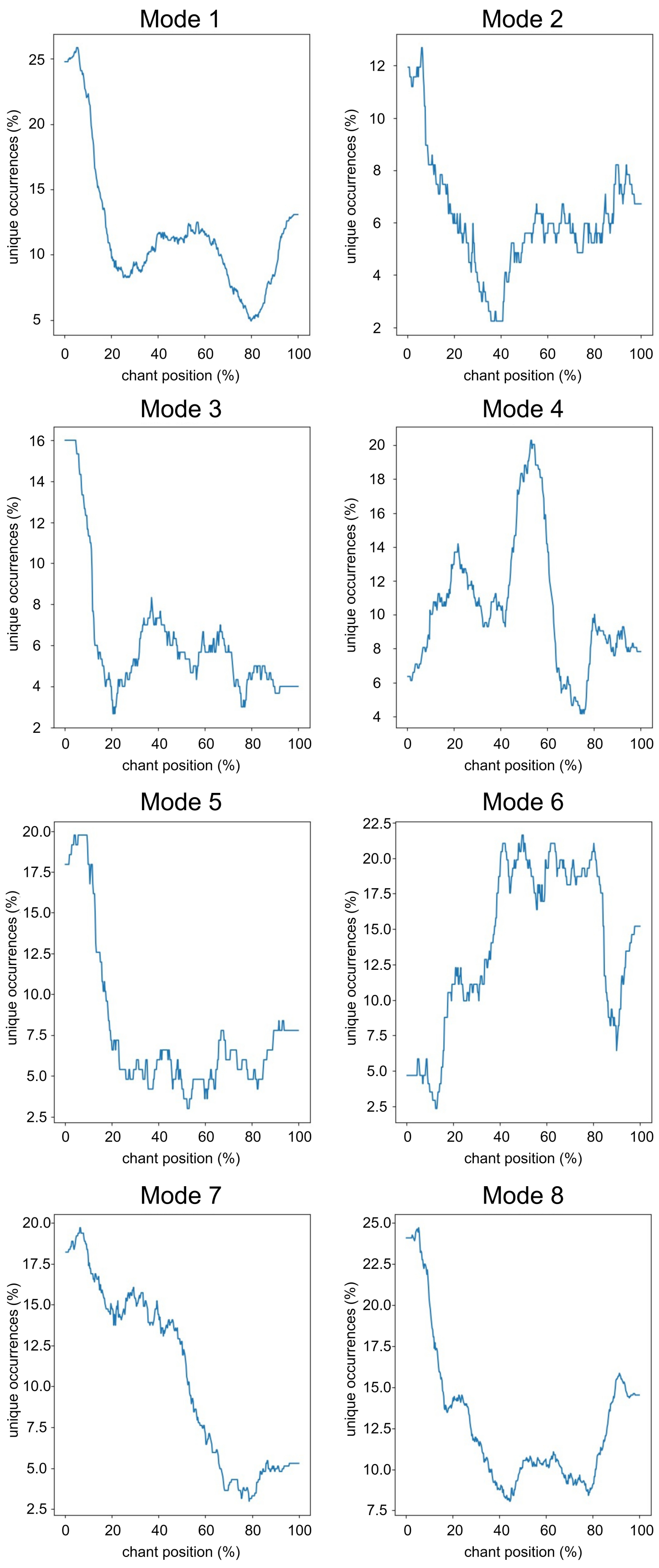


Figure 7.8: Antiphons-without-differentiae dataset - density of unique segments regarding each mode separately of the segmentation provided by the NHPYLM model. Each mode has a different chart curve, but we can notice that unique segments are placed mostly at the beginning.

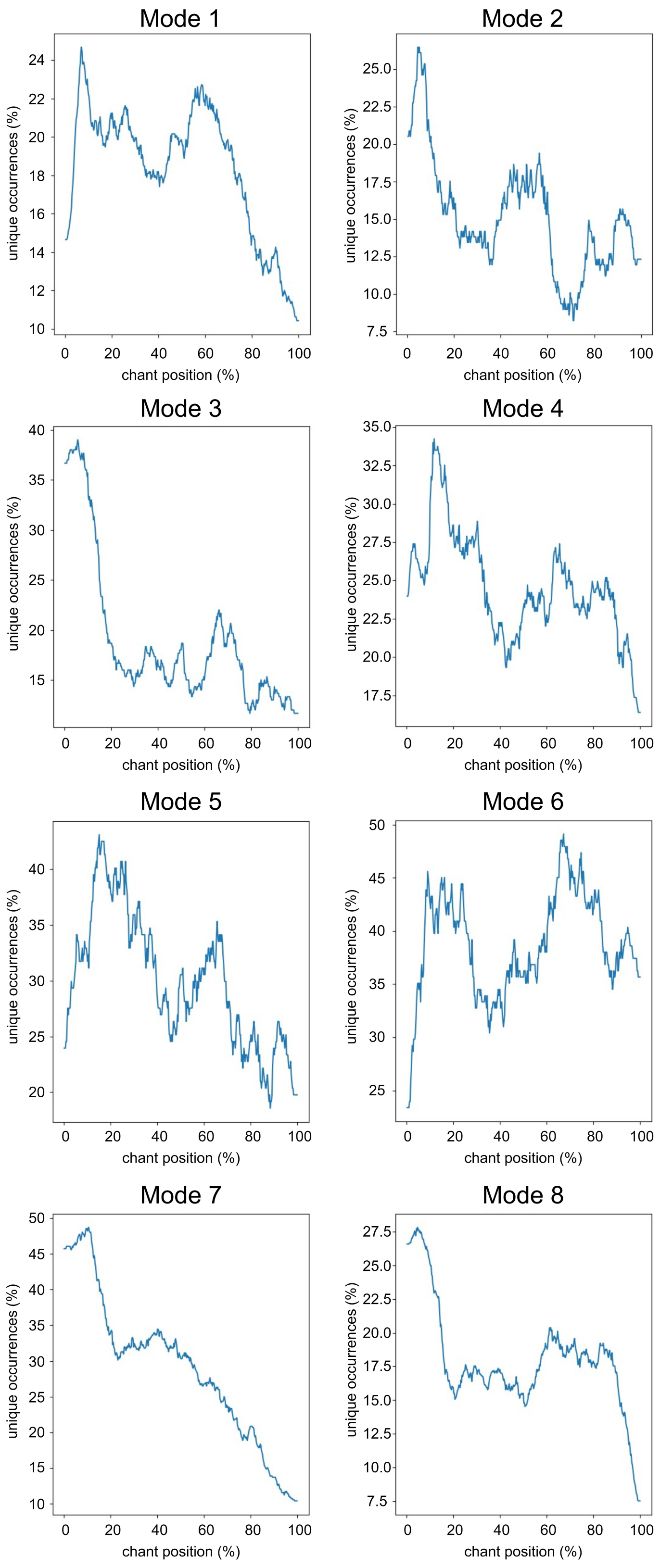


Figure 7.9: Antiphons-without-differentiae dataset - density of unique segments regarding each mode separately of the segmentation provided by the NHPYLM-Modes model. Charts show that NHPYLMModes found more unique segments than the NHPYLM model.

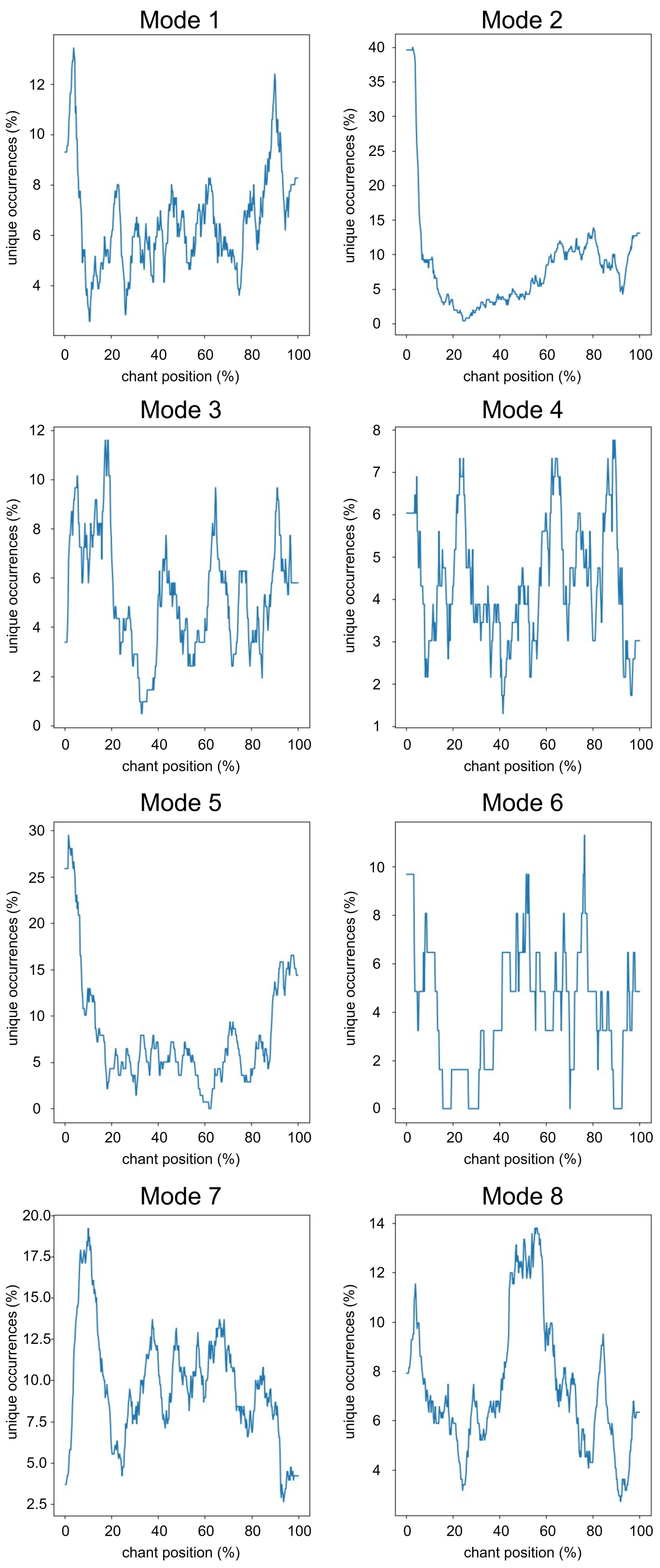


Figure 7.10: Responsory dataset - density of unique segments regarding each mode separately of the segmentation provided the by the NHPYLM model. Modes 2 and 5 have also a lot of unique segments at the beginning. The rest of modes does not contain many of unique segments.

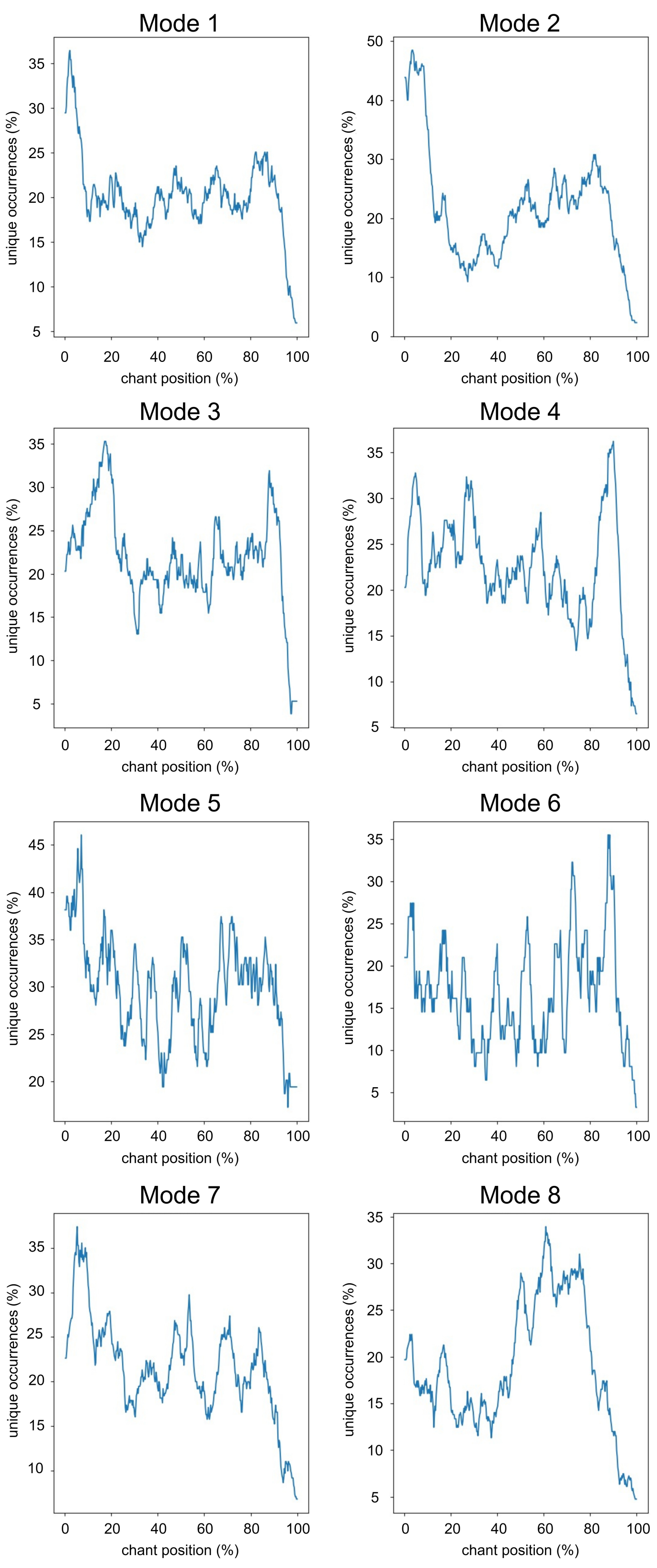


Figure 7.11: Responsory dataset - density of unique segments regarding each mode separately of the segmentation provided by the NHPYLMModes model. As the antiphons-without-differentiae dataset showed, the NHPYLMModes finds more unique segments than the NHPYLM model, but still, there is no common structure of the distribution of the unique segments over single modes.

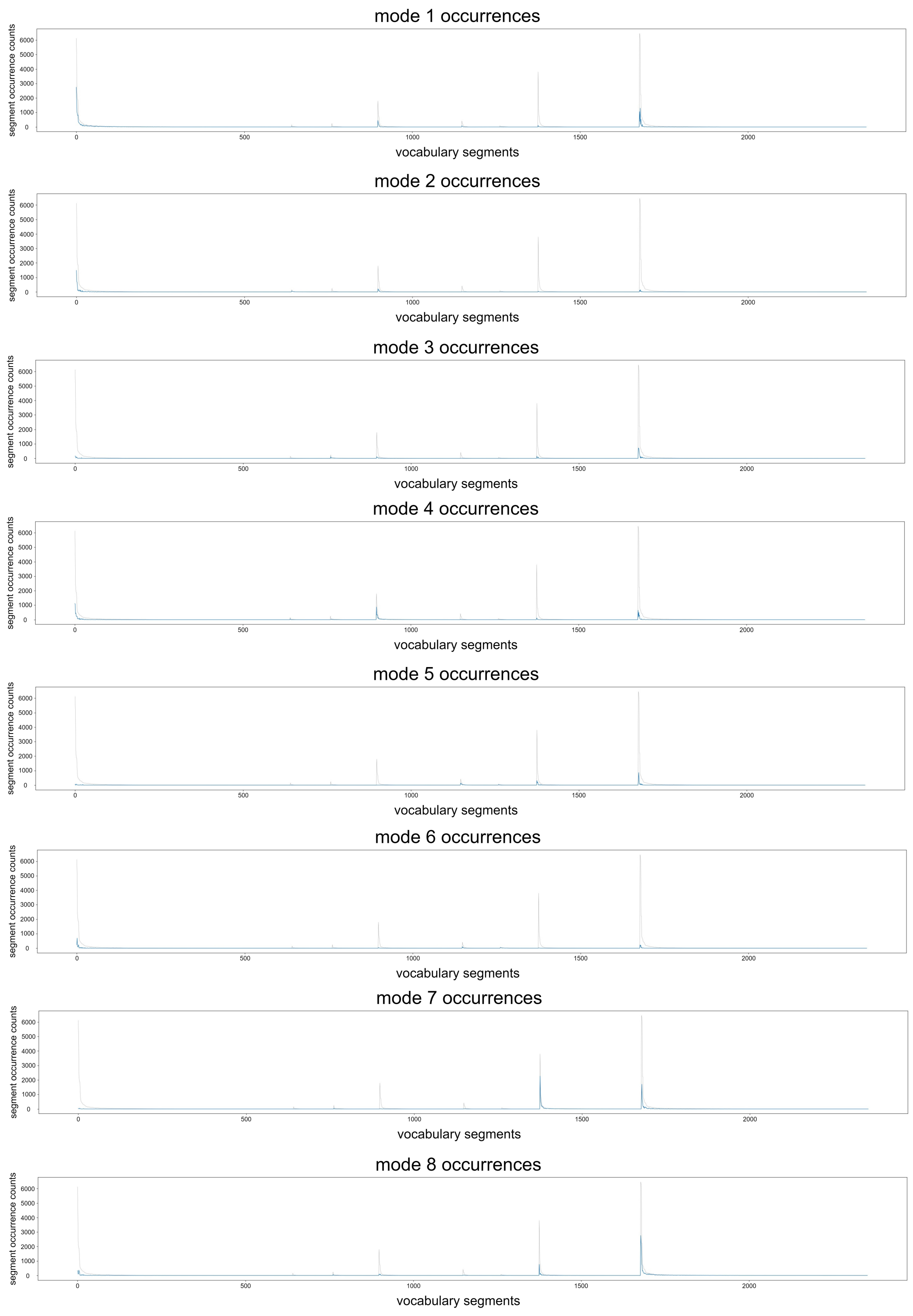


Figure 7.12: Antiphons-without-differentiae dataset - segments occurrences charts of each mode evaluated on the NHPYLM's segmentation. Note that the gray line is always common for all modes. Only the blue one differs. For instance, the seventh chart says that mode 7 uses segments associated with mode 7 and those associated with mode 8 similarly often. On the other hand, the eighth chart indicates that mode 8 primarily uses segments associated with mode 8 and only a few segments associated with mode 7. However, the model finds characteristic segments primarily of only four modes, the most frequent ones in the dataset. Many segments are shared not only among authentic-plagal mode pair.

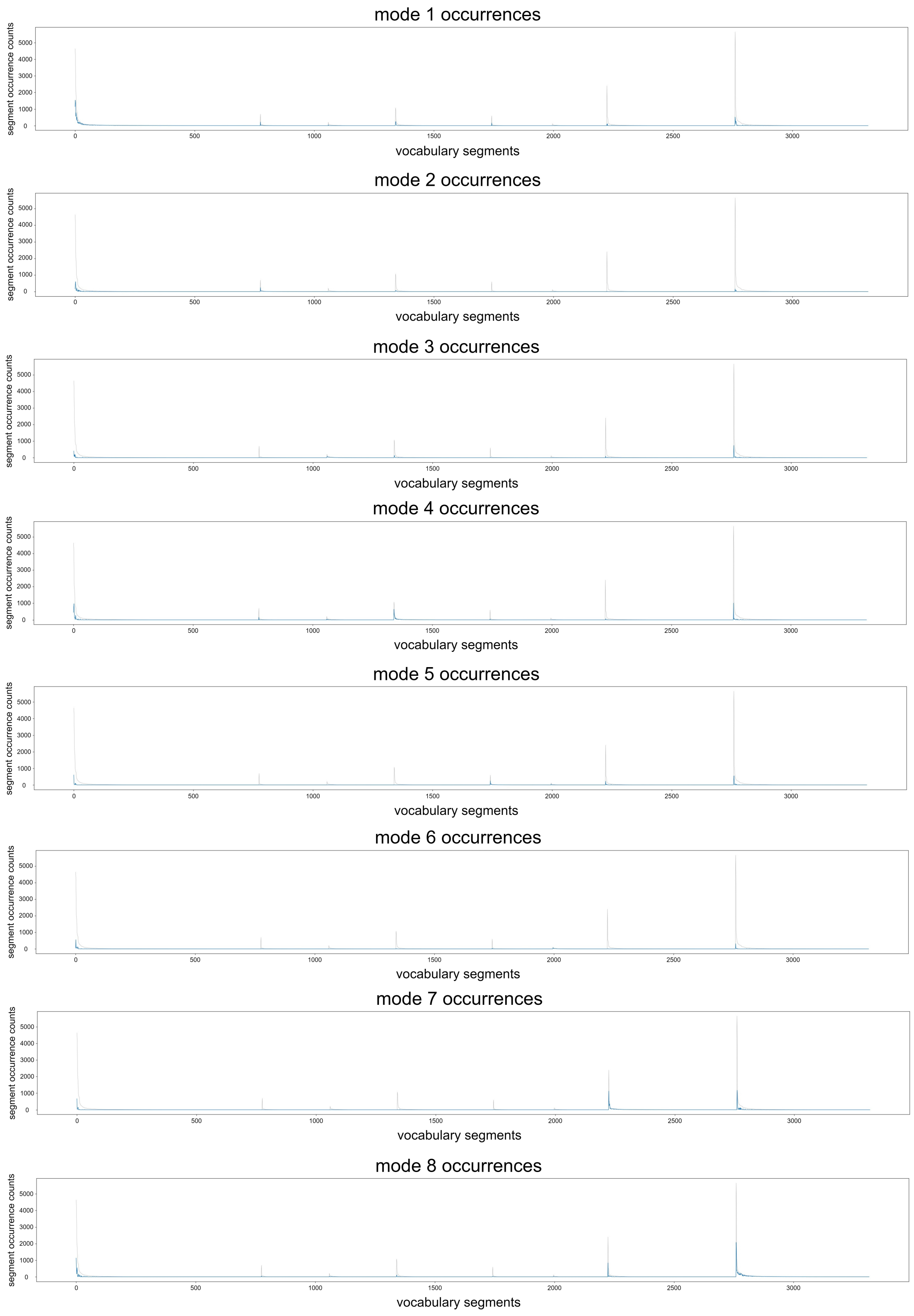


Figure 7.13: Antiphons-without-differentiae dataset - segments occurrences charts of each mode evaluated on the NHPYLMModes's segmentation. As peaks show, the model is able to find characteristic segments of all eight modes. Similarly to the NHPYLM model, there are a few frequent segments of each mode and many rare ones.

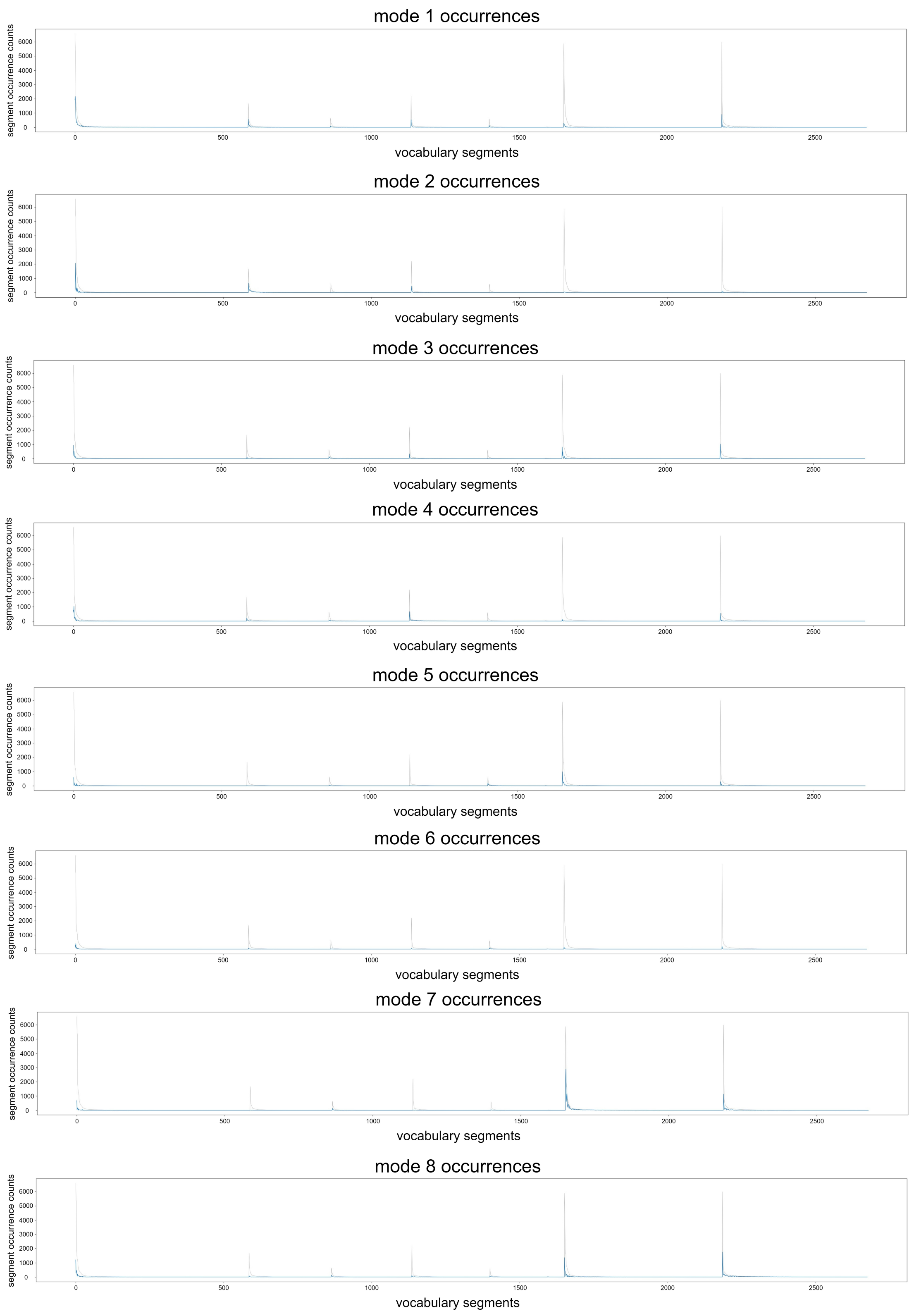


Figure 7.14: Responsory dataset - segments occurrences charts of each mode evaluated on the NHPYLM's segmentation. The behavior of the responsory and antiphon datasets is the same. In the case of the responsory dataset, NHPYLM model finds characteristic segments of most of the modes, which was an issue in the NHPYLM experiment of the antiphons-without-differentiae dataset.

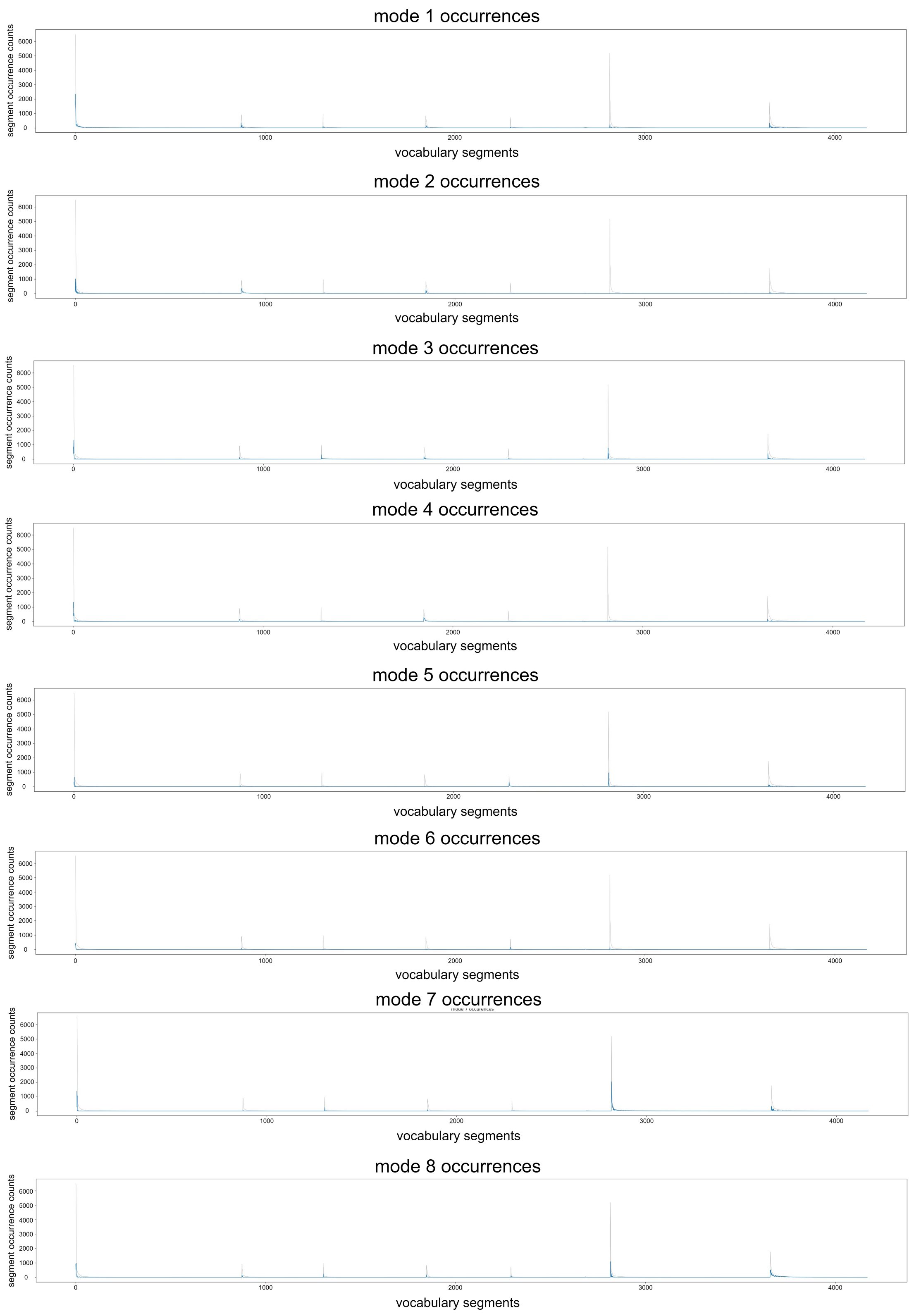


Figure 7.15: Responsory dataset - segments occurrences charts of each mode evaluated on the NHPYLMModes's segmentation. The numbers of characteristic segments of all modes are more even than in the case of the NHPYLM model. Again, there are few important melodic units, but also there are many rare segments that are not that interesting in the context of the memorization process.

7.2.3 The Impact of Modality Knowledge on Segmentation Process

As we can see from Tables 7.4 and 7.7, NHYPLMModes has similarly good accuracy and f1 scores using Naive Bayes (nb_accuracy, nb_f1), as the prediction given by SVC (bacor_accuracy, bacor_f1). This suggests that the segmentation by the NHPYLMModes model is also very well structured, and the difference between mode structures and their segments is not that complex. We can notice that the mode extension in the case of NHPYLM is a non-trivial improvement. This gives us another argument to support the theory that each mode has its own rules, properties, and structures of segments.

Again, it is essential to emphasize that in the case of NHPYLMModes or UMMs, we do not use any unallowed data, such as mode gold data of testing dataset, and so on. The unsupervised process, as well as the mode classification, is completely clean. During the testing, the model first predicts the mode using Bayes' theorem with segmentation probabilities provided by all eight submodels, as described in Section 6.2.5. Then the most probable segmentation generated by the submodel of the predicted mode is returned. How good is the internal mode classifier of the NHPYLMModes model? As we can see in Table 7.10, the ability of the model to predict the mode based on chant is similarly good as the bacor_accuracy, bacor_f1, nb_accuracy, and nb_f1 scores of the NHPYLMModes' segmentation. The internal model itself is robust enough to predict modes. It

	accuracy (%)
antiphons-without-differentiae	93.48
responsories	94.12

Table 7.10: Accuracies of internal mode classifiers of NHPYLMModes model on the testing dataset of both antiphons-without-differentiae and responsories.

only prepares the segmentation that could be easier processed by SVC and Naive Bayes classifiers. This is another argument to support that each mode and its segmentations work differently. Mistakes of the NHPYLMModes internal mode classifier are shown in the confusion matrices in Figure 7.16. The model is confused by pairs of authentic and its plagal modes that share many properties, such as final tones, etc.

To verify that the reason for better results of NHPYLMModes is because each mode was trained with a separate NHPYLM model and not just a product of having eight times as many parameters, we ran five more experiments for both datasets. We shuffled the training chants' correct modes with a random seed and used these pre-generated mode labels to assign the chants to their NHPYLM submodels. The experiment results are listed in Table 7.11 for antiphons, and Table 7.12 contains results for those experiments measured on responsories. Mode classification scores are suddenly worse by more than 1 % than the basic NHPYLM variant. This means that the fake NHPYLMModes model trained on shuffled gold data confuses the SVC and Naive Bayes classifiers. Furthermore, the scores of our shuffled experiments are even worse than nature segments proposed by Cornelissen et al. [2020b]. We can conclude that the improving factor of the NHPYLMModes model is the submodel separation.

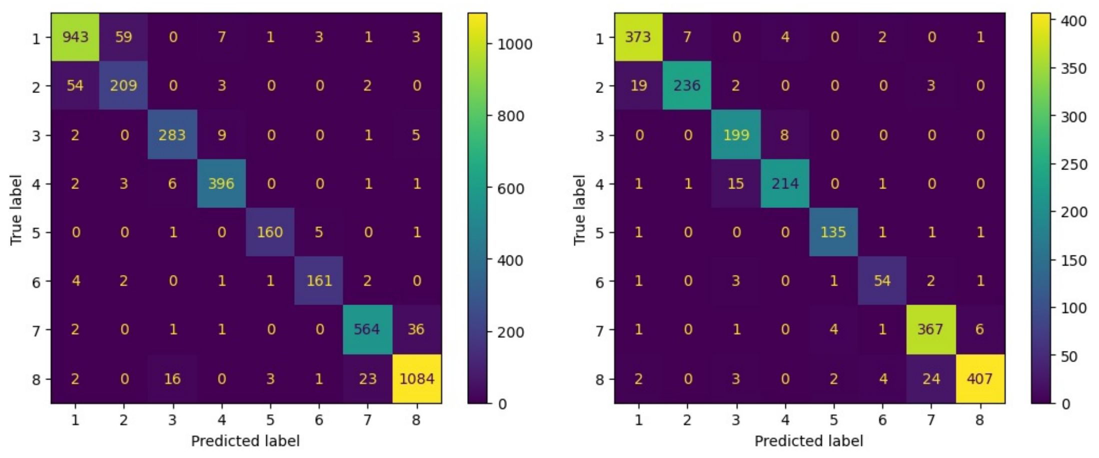


Figure 7.16: Confusion matrices of the NHPYLMModes internal mode classifier on the antiphons-without-differentiae dataset (left) and responsory dataset (right).

	bacor_accuracy	*bacor_f1*	*nb_accuracy*	*nb_f1*
seed=0	90.41	90.31	86.49	86.80
seed=1	90.55	90.46	85.31	85.75
seed=2	90.36	90.25	85.95	86.38
seed=3	90.80	90.71	86.77	87.09
seed=4	91.54	91.48	85.98	86.41
NHPYLM	*92.99*	*92.90*	*91.07*	*91.31*
NHPYLMModes	*94.02*	*94.01*	*93.58*	*93.59*

Table 7.11: Antiphons-without-differentiae dataset - mode classification scores of segmentations generated by fake NHPYLMModes using the shuffled mode labels on five different random seeds. Compared with the NHPYLM model with bacor_f1 score of 92.90, and the NHPYLMModes model trained on true mode labels with bacor_f1 score of 94.01, random mode labels only confuse the segmentation.

	bacor_accuracy	*bacor_f1*	*nb_accuracy*	*nb_f1*
seed=0	91.51	91.51	87.91	88.07
seed=1	91.51	91.49	88.24	88.35
seed=2	91.23	91.24	88.19	88.34
seed=3	91.37	91.37	88.62	88.78
seed=4	90.23	90.16	88.15	88.40
NHPYLM	*93.12*	*93.12*	*91.13*	*91.23*
NHPYLMModes	*94.22*	*94.22*	*94.22*	*94.21*

Table 7.12: Responsory dataset - mode classification scores of segmentations generated by fake NHPYLMModes using the shuffled mode labels on five different random seeds. Results are, on average, better than scores of the antiphons-without-differentiae shuffled experiment, but they are still not able to outperform the NHPYLM model nor the NHPYLMModes trained on the correct mode labels.

As we can see, the conditional non-parametric Bayesian model itself is a better mode classifier than the SVC classifier combined with the TF-IDF vectorizer. At the same time, we see that the segmentation of each mode behaves differently, i.e., each mode has such a different segmentation behavior that the model is able to recognize the mode of a new incoming chant purely by taking the most likely submodel that segments the melody most confidently. Thus, the knowledge of the mode helps in segmentation, and singers could use the information for better memorization of chant melodies.

7.3 Top Features

In this section, we will extract features and analyze their properties. We use the extraction method introduced in Section 5.4. From the discussions in previous sections, considering all our approaches, the NHPYLMModes provides the strongest segmentation. It performs best on the mode classification task and has the most confident and consistent segment properties over all modes listed in charts in Section 7.2.2. Also, the rest of the value score functions describe the segmentation provided by the NHPYLMModes as good segmentation. So we will use this segmentation to extract the top 100 features of both antiphons-without-differentiae and responsory datasets. Those extracted segments of the antiphons-without-differentiae dataset are listed in Table 7.13. Table 7.14 shows the top 100 features extracted from the responsory dataset. Segments are encoded into Volpiano notation as described in Section 1.5.1. There are short melodic units that occur many times in chants, but probably, they do not help to classify the mode that much. On the other hand, tables also consist of longer segments. Furthermore, on average, the top 100 segments of responsories are longer than those of antiphons.

g	k	h	f	d	l	hg	e	gh	gg
fe	fed	gf	efg	j	ff	hh	fg	df	c
ghg	kk	kj	lk	ll	i	fgh	cd	m	hk
dc	hgfg	ed	kjh	ghgf	cdd	fgg	dd	fh	fghg
hgh	fd	efgfedd	hhg	hjhgg	dcd	fghhgg	ggg	jk	kkjh
ee	fedd	jkl	fghhg	kjhg	hgg	cdf	kkj	hgf	hj
lml	hkh	de	fghh	lm	fgf	jh	fefg	ddcfg	kkl
hghg	hgfgg	cdfedd	gfed	ccd	ghgg	hjhg	ghk	hhgg	efgfed
fdc	defg	lkj	gfg	cddd	ki	jklk	fef	eg	dee
kh	fedcd	jkhg	fhk	ffg	efgg	klk	hkhg	ghgfg	lll

Table 7.13: Antiphons-without-differentiae dataset - top 100 features encoded in Volpiano tone notation. Features are ordered by their occurrence counts in the training dataset row by row from left to right. We can see the short segments as the most frequent ones, but also, some of the longer segments showed up to be important for the mode classification task.

We tested the strength of these segments by evaluating the mode classification scores on the chant segmentations, keeping only the top 100 features from Tables 7.13 and 7.14. The rest of the segments are removed for this evaluation purpose. The settings of the SVC used by Cornelissen et al. [2020b] that we use the same for bacor_accuracy and bacor_f1 score evaluation also use the hyperparameter tuning. The improvement of SVC performance is negligible. Moreover, the time

f	g	h	k	d	e	l	gh	j	hjkjhg
hg	gf	kj	c	fe	kl	gg	hjhghhg	hk	ghg
kk	fg	hhg	cd	i	lm	hh	fed	efg	df
dd	hkhghhg	ed	fgh	dc	jklkj	defefed	ff	kkj	ll
jk	fd	hgh	fh	nm	jh	fghg	ghgfggf	ffe	fgf
m	jkl	ghgg	klk	cdf	hgfg	hjh	dcd	gfgh	hkghg
ef	efd	cdd	hghg	eed	hgfghg	defed	hgf	ghkj	hghgfgg
efgfggf	gfg	efed	ln	fgfe	hjkjhgg	hih	gghg	fdf	lk
ghhg	llk	hkh	hgg	efedefd	fghh	egfffe	fhk	lkk	efede
fghghhg	dfd	hkk	ggg	defedcd	kjhghg	ggf	gfed	gff	klkj

Table 7.14: Responsory dataset - top 100 features encoded in Volpiano tone notation. We can notice that responsories have, on average, longer top 100 segments than antiphons. Another observation is that the top 100 features tables of both chant genres share 59 segments.

complexity is increased. Therefore, in this case, we ignore the tuning process. The final results of mode classification scores of chants segmentation with reduced segments are listed in Table 7.15.

	bacor_accuracy	*bacor_f1*	*nb_accuracy*	*nb_f1*
reduced antiphons	93.68	93.65	93.58	93.56
reduced responsories	94.12	94.12	94.22	94.22
full antiphons	*94.02*	*94.01*	*93.58*	*93.59*
full responsories	*94.22*	*94.22*	*94.22*	*94.21*

Table 7.15: Mode classification scores on chants segmentation originally generated by NHPYLMModes model, that their segments are reduced so the segmentation consists of only top 100 features from Tables 7.13 and 7.14 in case of antiphons-without-differentiae and responsory dataset respectively.

As we can see, scores of segmentation reduced on 100 segments is almost as good as the segmentation with all segments from the original vocabulary. Therefore, modes possess certain characteristic melodic units or their combinations that need to be identified in order to classify modes more accurately. Furthermore, memorizing 100 melodic units would not be that challenging anymore, so the existence of some compressing system is possible. Based on the information provided in Table 7.15, it is evident that these segments and their combinations are characteristic features for some of the modes. Also, charts in Figures 7.12 to 7.15 indicate that most of the vocabulary melodic units are rarely used in chants, and only a few are used very frequently. In order to visualize the percentage representation of top 100 segments in our best segmentation by NHPYLMModes model, and to observe the positions of these segments over all chants considering the specific mode, we generated charts of densities of top segments that are shown in Figures 7.17 and 7.18 of antiphons-without-differentiae and responsory datasets respectively.

Some of the modes have similar patterns. For instance, top segments occur more at the beginnings and endings than in the middle. Charts also show that these top segments occur very often in chants. Note that longer segments affect curves more significantly than shorter segments since longer segments are included

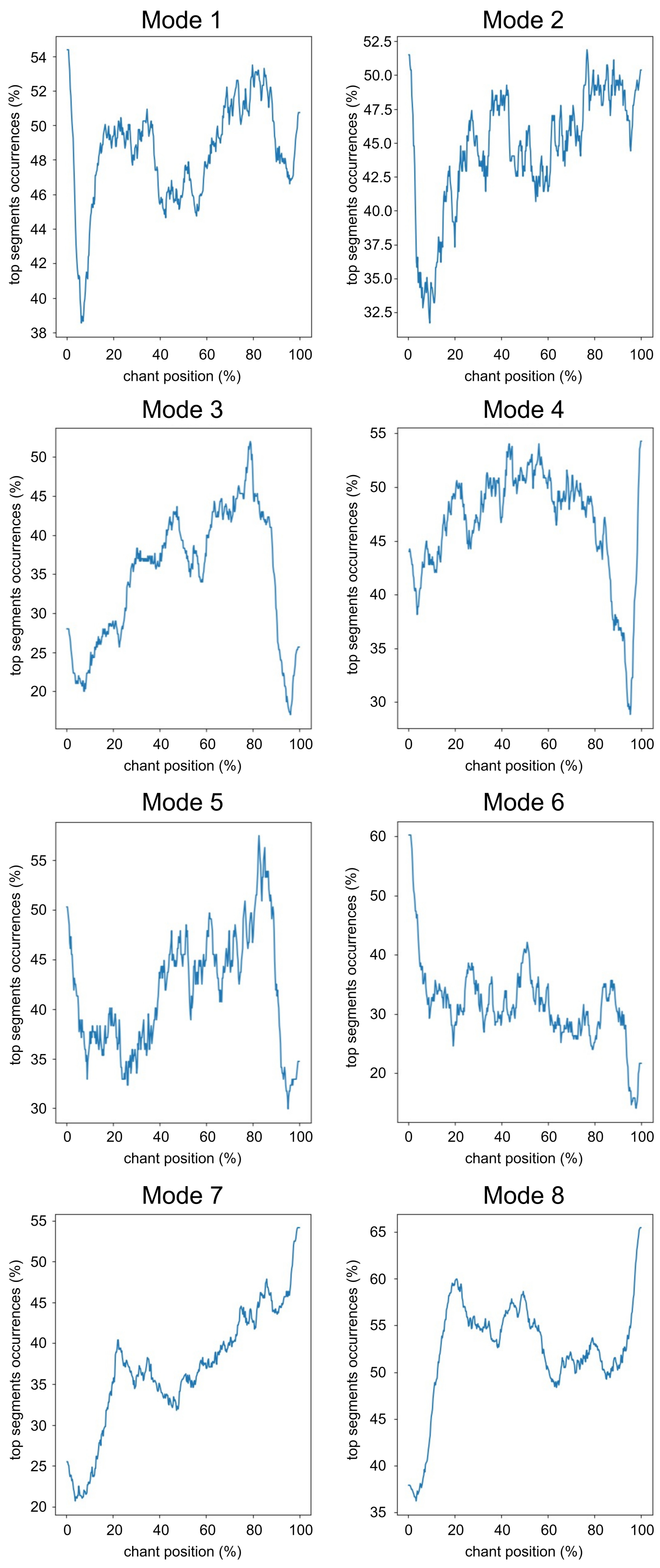


Figure 7.17: Antiphon-without-differentiae dataset - density of top segments over chants for each mode separately on segmentation provided by NHPYLMModes model. Charts show that there are no particular positions of top segments. On the other hand, top segments occur very often.

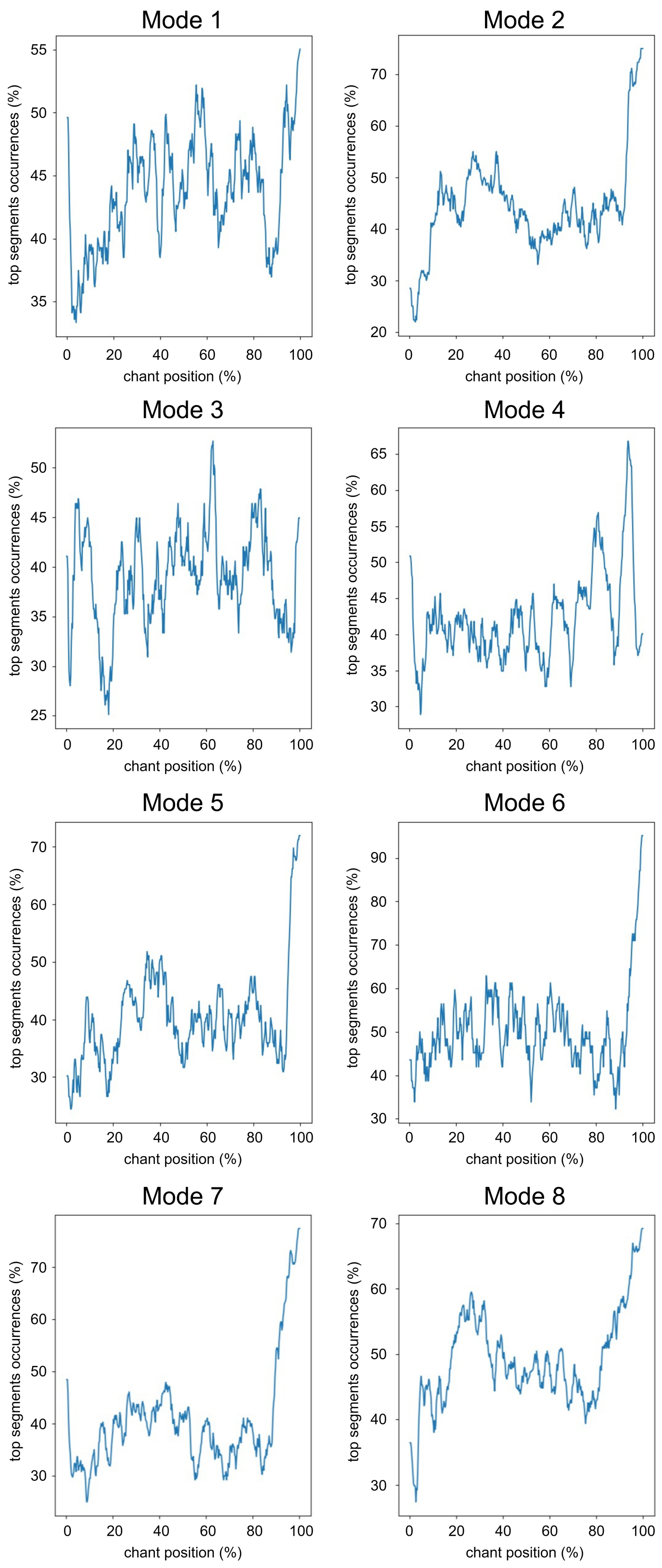


Figure 7.18: Responsory dataset - density of top segments over chants for each mode separately on segmentation provided by NHPYLMModes model. The top features of responsories are placed mostly at the end of chants, but they are also frequent in the rest of the melody positions.

in more chart bins (Section 5.3.3). But many of the top extracted features are short. Moreover, those short segments are the most frequent ones, where each of them influences a smaller area in the chart than those longer ones that are not frequent nor strong. Therefore, charts do not precisely tell the absolute frequency of all top segments occurrences over all segments.

The last analysis of the top segments we provide in this work examines the segment's dominant mode. Figure 7.19 contains two matrices of the top segments - modes analysis described in Section 5.4. The top one corresponds to our best segmentation of the antiphon-without-differentiae testing dataset. The bottom one corresponds to the responsory testing dataset. Each row corresponds to one of the modes: the top one is mode 8, the one below is mode 7, and in descending order from top to bottom row of mode 1. Each column is mapped to one of the top 100 features in the same order as segments are listed in Tables 7.13 for antiphons, and 7.14 for responsories, row by row. So the first ten columns of Figure 7.19 are mapped to the segments in the first row of Tables 7.13, or 7.14, in the same order. The following ten columns of Figure 7.19 are mapped to the second row in the same order, etc. In general, segments of length one, two, or three are more likely shared among all modes. On the other hand, longer segments are characteristic mostly for one specific mode.

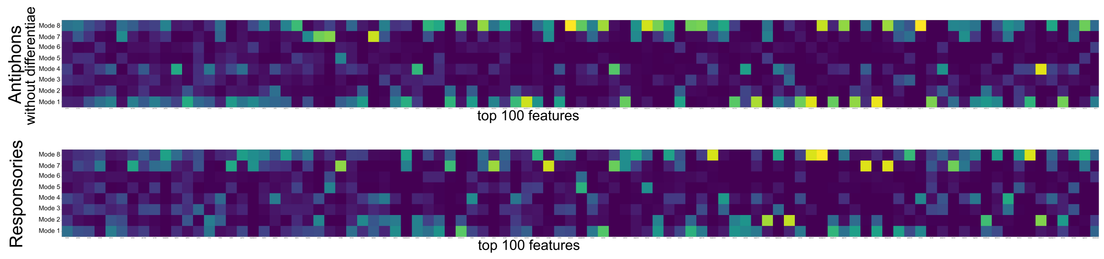


Figure 7.19: Occurence matrix of top features regarding modes. The top matrix corresponds to the segmentation of the antiphons-without-differentiae dataset. The bottom matrix corresponds to the segmentation of the responsory dataset. Both segmentations were generated by the NHPYLMModes model. We can see that all modes use the short top segments. On the other hand, those longer ones are used mostly in one specific mode.

7.4 Trimming Experiments

So far, there is much evidence that segments from the beginnings and ends of chants are more important in some way than those in the middle. Therefore, we do the trimming experiment on our best segmentations generated by the NHPYLMModes model. We remove an even number k of segments from chant segmentation, and we measure the resulting segmentation's bacor_accuracy and bacor_f1 scores, again without the parameter tuning. We want to measure the importance and strength of these marginal segments. We include four types of segment removal, and we compare them together. The first is the baseline of randomly removed k segments from the segmentation. Then we remove k segments from the left side of the chant, which is the beginning, and we also examine segment removal from the right side - the end of chants. The last approach removes segments from both sides. The first half of the segments are removed from the beginning, while the rest are removed from the end of the chant, which is the reason for k to be an even number. Results of the antiphons-without-differentiae experiment are visualized in Figure 7.20, where we removed zero to twelve segments. Figure 7.21 visualizes the results of the responsory experiment that includes segment removals of zero to thirty segments.

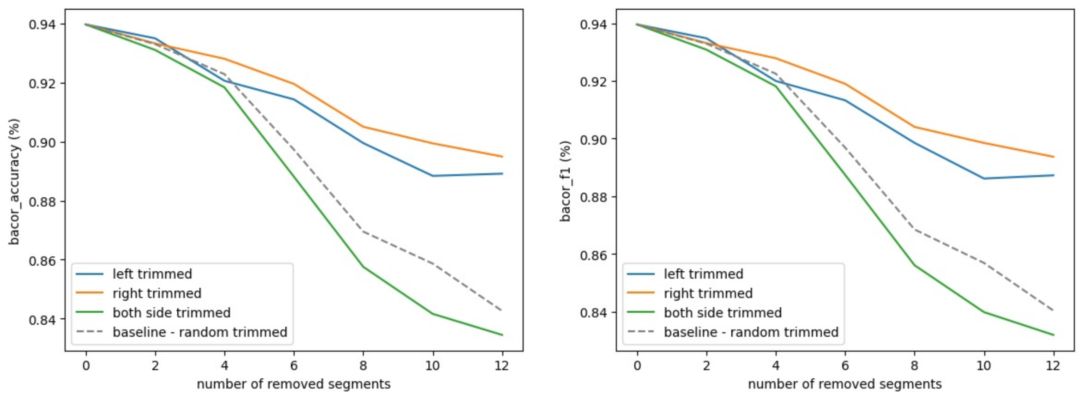


Figure 7.20: Antiphons-without-differentiae dataset - trimmed experiment on the segmentation provided by NHPYLMModes model. The left chart shows the bacor_accuracy, and the right chart shows the bacor_f1 scores. The combination of beginning and final segments is stronger than segments from only one chant side.

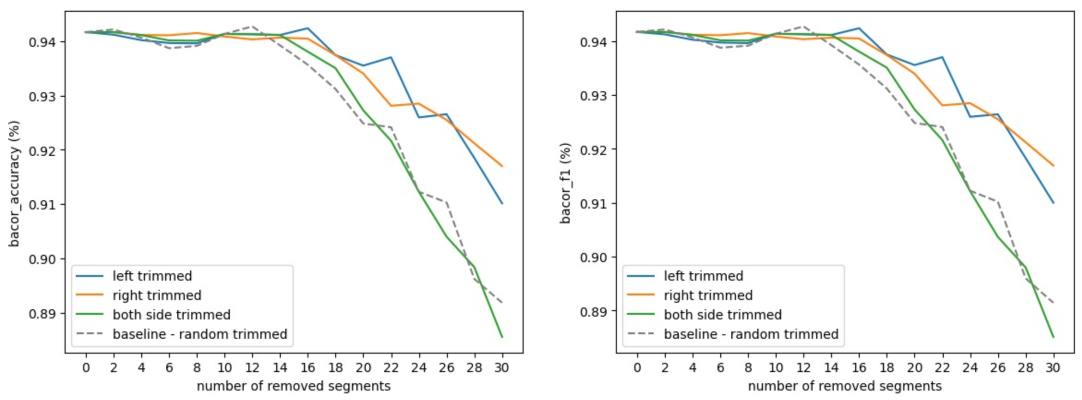


Figure 7.21: Responsory dataset - trimmed experiment on the segmentation provided by NHPYLMModes model. The left chart shows the bacor_accuracy, and the right chart shows the bacor_f1 scores. As in the case of antiphons, the combination of the responsory beginning and final segments is also stronger than segments from the beginning or segments from the end of chants. Surprisingly, the combination is similarly strong as randomly chosen segments, which are also better than segments from only one side of the chant. Note that the first 6 random segments were the weakest ones.

The results show that antiphons have more informative beginnings than endings, which cannot be said with certainty about responsories. Let's compare removing segments from the front or the end of the chant with a random baseline where we randomly select segments to remove. We find that the random removal of segments affects the results more significantly. Removing segments from both sides at the same time (when the number of removed segments is still the same as considering other approaches) is the most influential in both dataset cases. The combination of beginning and ending segments could be crucial for mode classification. On the other hand, it could be possibly explained by the fact that when we remove the segment only from one side, the mode could still be predicted using information from the other side.

Conclusions and Future Work

Gregorian chant, as an oral tradition, was performed by singers who learned thousands of melodies for the entire liturgical year. There are hypotheses suggesting that one of the possible ways how singers could remember the repertoire is based on the melody segmentation and its melodic units. Therefore, in this work, we dealt with the unsupervised segmentation of Gregorian chants of the two most frequent genres of chants - antiphons and responsories. We were looking for a system of melodic units that would simplify chant melodies' memorization and could also relate to chant modes. We designed several evaluation functions and visualizations to analyze predicted segmentations. Some of the score functions and metrics were focused on modal properties, while others observed statistical aspects of the model and its segmentation. Then, we introduced three segmentation models we used in this work. The first was a baseline unigram model. The second was the NHPYLM model based on Pitman-Yor processes which showed up to perform best on this task. On the other hand, the third one, based on BERT, is unsuitable for our task, either for the complex structure of chants or for the small amount of training data. The best invention of this work was the extension of the NHPYLM model that trains each mode separately. Here we summarize our findings.

Natural segmentation by words or syllables is not ideal. Our work partially follows up on the article by Cornelissen et al. [2020b], which says that the best segmentation is the one that is based on natural units such as words or syllables. We showed that they had used the antiphon data incorrectly, and their results needed to be reevaluated. Furthermore, Tables 7.1 to 7.9 show that we found the segmentation with not only a higher mode classification score than the segmentation based on natural units but also with other properties that indicate that our segmentation is more fitting to the Gregorian chant context. Our proposed segmentation also gives a mode classification score very close to our proposed upper bound for segmentation-based mode classification, i.e., overlapping n-grams. Considering this overlapping n-grams approach, we set the state-of-the-art on the mode classification task.

The beginnings and ends of chants have stronger modal identity than the segments in the middle. Based on Figures 7.2 to 7.5 we discovered that segments at the beginning and especially at the end of chants have on average longer segments. This means that the tones at the beginning and the end are more related to each other, and therefore the resulting segments of these tones are more stable. As Figures 7.8 to 7.11 show, there are more unique melodic units that are used only by one mode at the beginning of chants than in the middle or at the end. These unique segments determine the mode of the chant melody. We can also notice that in the context of the average segment length and the segment uniqueness, each mode behaves differently, which could be caused by the small amount of training data, but also by the different rules and properties of single modes. However, combining the results of these two types of charts, the beginning and ending segments have a stronger modal identity than the rest of the chant. This is also supported by the trimmed experiment visualized in Figures 7.20 and 7.21. This observation lends further weight to the view of modality as a phenomenon primarily of performance practice rather than medieval music theory.

Many segments are shared among all modes, but there are also segments that are used by only one mode. We also analyzed that many melodic units are shared among all modes, especially when considering the authentic-plagal mode pairs, as shown in Figures 7.6, 7.7 and 7.12 to 7.15. But still, based on the *wtmf* score from Tables 7.3, 7.6 and 7.9, melodic units are used primarily by one particular mode. Furthermore, as Figures 7.6 to 7.11 show, some of the melodic units are used only by one specific mode. Again, this can be taken to imply that modality is not necessarily a principle governing the entire melody, but more detailed musicological work is necessary.

Conditional NHPYLM model generates the segmentation that gives the state-of-the-art performance on the mode classification task based on the melody segmentation. We proposed the mode extension of the NHPYLM model that contains eight separated NHPYLM submodels trained individually. Tables 7.1 to 7.9 show that each NHPYLM submodel is trained separately well to segment the chant, but based on the Table 7.10 and Figure 7.16, these eight submodels are also trained well to recognize chants that belong to the mode of the particular submodel. The internal mode classifier of the extended NHPYLM model is more accurate than any of the previously proposed methods by us or Cornelissen et al. [2020b]. In case all modes have the same segmentation properties, the internal mode classifier would be random. Therefore, each mode has its own independent segmentation properties that allow training each NHPYLM submodel to be able to recognize the chant of the submodel's mode. Therefore, the behavior in a way of compounding a single tone into segments of each mode is unique. Applying the TF-IDF vectorizer with the SVC classifier on the predicted segmentation using the extended NHPYLM model, we reach the state-of-the-art on the mode classification task based on melody segmentation.

There are only a few frequent segments and a lot of occasional ones. Charts in Figures 7.12 to 7.15 imply that the segmentation contains only a few frequent segments, but most of the vocabulary segments have occurred rarely. As part of this work, we listed the top 100 extracted segments encoded in Volpiano notation for both types of chants in Tables 7.13 and 7.14. From Figures 7.17 and 7.18, it could be seen that we chose frequent segments. Antiphons share 59 of top segments with responsories. Therefore, it is possible that the base melodic units strongly indicative of mode don't relate to a particular chant genre. However, based on Table 7.15, these top segments are powerful enough to obtain similar scores in the mode classification task as the original segmentation that uses all segments from the actual vocabulary. Compressing melodies with these 100 segments is efficient despite the large number of occurrences of single-tone segments. On the other hand, it still does not seem practical for singers to think of melodies in this way during the learning process.

Modality and segmentation are closely related because if we use a model where we assume that each mode segmentation behaves differently, we get better performance on mode classification. Also, peaks of average segment length, density of unique segments, or segment occurrences charts exhibit greater clarity and confidence of all modes. Hence, the mode-dependent model also learns properties of modes whose chants are less frequent in the dataset. Furthermore, each mode has its unique segments but also shares segments with others. It's not surprising that the beginnings and endings of chants have a stronger modal identity than the rest since chants are linked by these segments together in the repertoire. The segmentation we found contains many single-tone or rare segments, indicating that it cannot be directly segmented into fixed melodic units. Therefore we demonstrate another perspective showing that centonization is likely not a good model of chant melody. We know that the mode of the chant influences the melody segmentation. We also know that melodies contain a lot of commonly repeated melodic units, and the vocabulary of those frequent melodic units is not that large. This could mean that these frequent melodic units just settled down naturally over time because they were well remembered and sung, and similar melodic units were transformed into them. On the other hand, these melodic units are too short for that. But also, it could be caused by segment development and modification over the years, so the segmentation contains a lot of similar but still different melodic units. Alternatively, considering that we found relatively small melodic units that strongly determine the chant modes, the Gregorian chant tradition could be based on a system of particular grammar and its rewriting rules, where we found terminals and their most frequently used compounds. Based on our observations, we are not able to provide a precise explanation of the chant learning and memorization processes, but as we discussed in the introduction, the work does not have the goal of finding breakthrough discovery in the area of the Gregorian chant methodology. A further meaning of our measurements has to be found in close collaboration with chant scholars and experts on medieval music theory.

In future work, the statistical independence of modes and their segmentations needs to be proven. We also haven't analyzed the explicit efficiency of compressing chant melodies by memorizing top segments. In other words, if each of the top segments were replaced by one sign indicating the segment, how much data that singers had to memorize would be reduced? Furthermore, it is not clear how top segments evolved over the centuries and whether those not frequent segments are similar to those in the list of top 100 features. It is essential to design a segmentation model that considers similar melodic units as one and measure its performance. It is also worth analyzing melodies as a result of some grammar and its rewriting rules on the top segments proposed by us.

Given the overall rough agreement between existing musicological knowledge and the behavior of our segmentation models, we believe that this is a viable avenue of digital chant research, but a closer collaboration with chant experts is needed to design models. Although there are still many unanswered questions, our work has described not only why it is worth addressing this topic but also in which direction it is intriguing to proceed further.

Bibliography

Berke Akkaya and Nurdan Çolakoğlu. Comparison of multi-class classification algorithms on early diagnosis of heart diseases. In *Proceeding Book of the y-BIS Conference 2019: Recent Advances in Data Science and Business Analytics*, September 2019.

Doug Beeferman, Adam Berger, and John Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1):177–210, February 1999. ISSN 1573-0565. doi: 10.1023/A:1007506220214. URL `https://doi.org/10.1023/A:1007506220214`.

Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

Olivier Berten. A database of gregorian scores, 2013. URL `https://gregobase.selapa.net/`.

Christopher Bishop. *Pattern Recognition and Machine Learning.* Springer, January 2006. URL `https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/`.

David Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. ISSN 1532-4435. doi: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. URL `http://portal.acm.org/citation.cfm?id=944937`.

Matt Bognar. Gamma distribution, 2021. URL `https://homepage.divms.uiowa.edu/~mbognar/applets/gamma.html`. [Accessed 11-Jun-2023].

Bas Cornelissen. gabc2volpiano, January 2020. URL `https://github.com/bacor/gabc2volpiano`.

Bas Cornelissen, Willem Zuidema, and John Ashley Burgoyne. Studying large plainchant corpora using chant21. In *7th International Conference on Digital Libraries for Musicology*, DLfM 2020, page 40–44, New York, NY, USA, 2020a. Association for Computing Machinery. ISBN 9781450387606. doi: 10.1145/3424911.3425514. URL `https://doi.org/10.1145/3424911.3425514`.

Bas Cornelissen, Willem Zuidema, and John Ashley Burgoyne. Mode Classification and Natural Units in Plainchant. *Proceedings of the 21st ISMIR Conference*, pages 869–875, 2020b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

Jacob Eisenstein and Regina Barzilay. Bayesian unsupervised topic segmentation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 334–343, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL `https://aclanthology.org/D08-1035`.

Paolo Ferretti. *Estetica gregoriana: ossia, Trattato delle forme musicali del canto gregoriano*. Number sv. 1 in Estetica gregoriana: ossia, Trattato delle forme musicali del canto gregoriano. Pontificio istituto di musica sacra, 1934. URL `https://books.google.cz/books?id=vOWCnQEACAAJ`.

Daniel Fink. A compendium of conjugate priors. 46, 1997.

G David Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

Yixing Guan, Jinyu Zhao, Yiqin Qiu, Zheng Zhang, and Gus Xia. Melodic phrase segmentation by deep neural networks, 2018.

Katherine Eve Helsen. *The Great Responsories of the Divine Office: aspects of structure and transmission*. PhD thesis, University of Regensburg, 2008.

David Hiley. *Western Plainchant: A Handbook*. Clarendon Press, 1993. ISBN 9780198162896. URL `https://books.google.cz/books?id=9-QXAQAAIAAJ`.

David Hiley. *Contents*, pages vii–viii. Cambridge Introductions to Music. Cambridge University Press, 2009.

John D Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Sayali D Jadhav and HP Channe. Comparative study of k-nn, naive bayes and decision tree classification techniques. *International Journal of Science and Research (IJSR)*, 5(1):1842–1845, 2016.

Debra Lacoste. The cantus database and cantus index network. In *The Oxford Handbook of Music and Corpus Studies*. Oxford University Press, 2022. ISBN 9780190945442. doi: 10.1093/oxfordhb/9780190945442.013.18. URL `https://doi.org/10.1093/oxfordhb/9780190945442.013.18`.

Stefan Lattner, Carlos Eduardo Cancino Chacón, and Maarten Grachten. Pseudo-supervised training improves unsupervised melody segmentation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Fred Lerdahl and Ray Jackendoff. *A generative theory of tonal music*. The MIT Press, Cambridge. MA, 1983. ISBN 0262120941.

Kenneth Levy. The italian neophytes' chants. *Journal of the American Musicological Society*, 23(2):181–227, 1970.

Wei Li, Yuhan Song, Qi Su, and Yanqiu Shao. Unsupervised Chinese word segmentation with BERT oriented probing and transformation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3935–3940, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi:

10.18653/v1/2022.findings-acl.310. URL `https://aclanthology.org/2022.findings-acl.310`.

David JC MacKay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

Massimo Melucci and Nicola Orio. Evaluating automatic melody segmentation aimed at music information retrieval. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 310–311, 2002.

Daichi Mochihashi and Eiichiro Sumita. The infinite markov model. *Advances in neural information processing systems*, 20, 2007.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 100–108, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL `https://aclanthology.org/P09-1012`.

Giorgos Mountrakis, Jungho Im, and Caesar Ogole. Support vector machines in remote sensing: A review. *ISPRS journal of photogrammetry and remote sensing*, 66(3):247–259, 2011.

Thomas Nuttall, Miguel García Casado, Víctor Núñez Tarifa, Rafael Caro Repetto, and Xavier Serra. Contributing to new musicological theories with computational methods: the case of centonization in arab-andalusian music. In *20th Conference of the International Society for Music Information Retrieval (ISMIR 2019): 2019 Nov 4-8; Delft, The Netherlands.[Canada]: ISMIR; 2019. p. 223-8.* International Society for Music Information Retrieval (ISMIR), 2019.

Peter Orbanz and Yee Whye Teh. Bayesian nonparametric models. *Encyclopedia of machine learning*, 1, 2010.

Marcus T. Pearce, Daniel Müllensiefen, and Geraint A. Wiggins. *Melodic Grouping in Music Information Retrieval: New Methods and Applications*, pages 364–388. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-11674-2. doi: 10.1007/978-3-642-11674-2_16. URL `https://doi.org/10.1007/978-3-642-11674-2_16`.

Juan Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

Shun Sawada, Kazuyoshi Yoshii, and Keiji Hirata. Unsupervised melody segmentation based on a nested Pitman-Yor language model. In *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, pages 59–63, Online, 16 October 2020. Association for Computational Linguistics. URL `https://aclanthology.org/2020.nlp4musa-1.12`.

Alessandro Solbiati, Kevin Heffernan, Georgios Damaskinos, Shivani Poddar, Shubham Modi, and Jacques Calì. Unsupervised topic segmentation of meetings with BERT embeddings. *CoRR*, abs/2106.12978, 2021. URL `https://arxiv.org/abs/2106.12978`.

Barbara Swanson, Jennifer Bain, Kate Helsen, Jan Koláček, Debra Lacoste, and Alessandra Ignesti. Volpiano protocols with self-test, 2016. URL `https://cantus.uwaterloo.ca/sites/default/files/documents/2.%20Volpiano%20Protocols.pdf`. Accessed on 05.21.2023.

Yee Whye Teh. A bayesian interpretation of interpolated kneser-ney nus school of computing technical report tra2/06. *National University of Singapore*, pages 1–21, 2006a.

Yee Whye Teh. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, 2006b.

Yee Whye Teh and Michael I Jordan. Hierarchical bayesian nonparametric models with applications. *Bayesian nonparametrics*, 1:158–207, 2010.

Tonary of Regino of Prüm. Brussels 2750/65. Bibliothèque Royale de Belgique, 10AD.

Leo Treitler. Centonate chant: Centonate or centonate. *J. Am. Music. Soc.*, 28 (1):1–23, April 1975.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Linzi Xing and Giuseppe Carenini. Improving unsupervised dialogue topic segmentation with utterance-pair coherence scoring, 2021.

Mohammad Zeineldeen. Hidden markov model, 7 October 2018. URL `https://mmz33.github.io/Hidden-Markov-Model/`.

List of Figures

List of Tables

List of Abbreviations

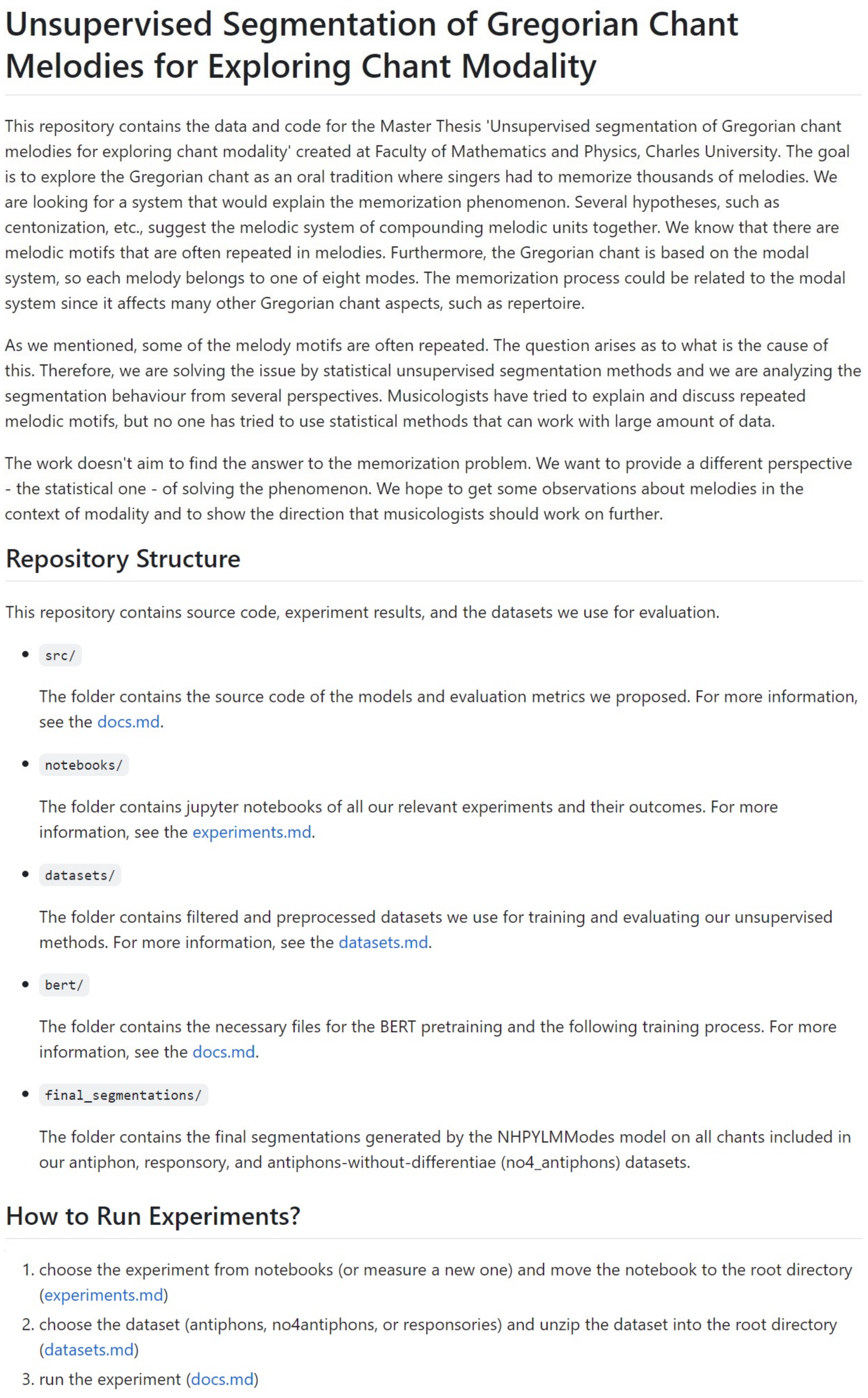
TF-IDF	Term Frequency-Inverse Document Frequency
SVC	Support Vector Classifier
BERT	Bidirectional Encoder Representations from Transformers
RoBERTa	A Robustly Optimized BERT Pretraining Approach
HPYLM	hierarchical Pitman-Yor language model
IDyOM	Information Dynamics of Music
RBM	Restricted Boltzmann Machine
CNN-CRF	convolutional neural networks - conditional random field
HMM	hidden Markov model
NHPYLM	nested hierarchical Pitman-Yor language model
SHPYLM	segment hierarchical Pitman-Yor language model
THPYLM	tone hierarchical Pitman-Yor language model

A. Attachments

A.1 GitHub Repository

`https://github.com/lanzv/Master-Thesis`

A.2 README.md



Unsupervised Segmentation of Gregorian Chant Melodies for Exploring Chant Modality

This repository contains the data and code for the Master Thesis 'Unsupervised segmentation of Gregorian chant melodies for exploring chant modality' created at Faculty of Mathematics and Physics, Charles University. The goal is to explore the Gregorian chant as an oral tradition where singers had to memorize thousands of melodies. We are looking for a system that would explain the memorization phenomenon. Several hypotheses, such as centonization, etc., suggest the melodic system of compounding melodic units together. We know that there are melodic motifs that are often repeated in melodies. Furthermore, the Gregorian chant is based on the modal system, so each melody belongs to one of eight modes. The memorization process could be related to the modal system since it affects many other Gregorian chant aspects, such as repertoire.

As we mentioned, some of the melody motifs are often repeated. The question arises as to what is the cause of this. Therefore, we are solving the issue by statistical unsupervised segmentation methods and we are analyzing the segmentation behaviour from several perspectives. Musicologists have tried to explain and discuss repeated melodic motifs, but no one has tried to use statistical methods that can work with large amount of data.

The work doesn't aim to find the answer to the memorization problem. We want to provide a different perspective - the statistical one - of solving the phenomenon. We hope to get some observations about melodies in the context of modality and to show the direction that musicologists should work on further.

Repository Structure

This repository contains source code, experiment results, and the datasets we use for evaluation.

- `src/`

  The folder contains the source code of the models and evaluation metrics we proposed. For more information, see the docs.md.

- `notebooks/`

  The folder contains jupyter notebooks of all our relevant experiments and their outcomes. For more information, see the experiments.md.

- `datasets/`

  The folder contains filtered and preprocessed datasets we use for training and evaluating our unsupervised methods. For more information, see the datasets.md.

- `bert/`

  The folder contains the necessary files for the BERT pretraining and the following training process. For more information, see the docs.md.

- `final_segmentations/`

  The folder contains the final segmentations generated by the NHPYLMModes model on all chants included in our antiphon, responsory, and antiphons-without-differentiae (no4_antiphons) datasets.

How to Run Experiments?

1. choose the experiment from notebooks (or measure a new one) and move the notebook to the root directory (experiments.md)
2. choose the dataset (antiphons, no4antiphons, or responsories) and unzip the dataset into the root directory (datasets.md)
3. run the experiment (docs.md)

A.3 docs.md

Documentation

We propose several unsupervised segmentation models and evaluation functions that analyze the predicted segmentation. All is implemented in Python (eventually Cython) and placed in the `src/` directory. The directory is structured this way:

- `src/`
  - `eval/`
    - evaluating segmentation on our score functions
    - evaluation pipelines
    - script to do the (top 100) feature extraction
  - `models/`
    - unigram models
    - NHPYLM models
    - BERT
    - random baseline
    - overlapping n-grams upper bound
  - `utils/`
    - plotters
    - data loaders (datasets.md)
    - training iteration statistics
    - dataset analysis
    - gregobase preparation

Models - Usage

First of all, install all necessary packages:

```
pip install -r requirements.txt
```

UM (Unigram Model)

Suppose we have list of string melodies `X_train`, `X_dev`, and `X_test` and their corresponding modes `y_train`, `y_dev`. Then we can get the test segmentation, perplexity, and mawp score this way:

```python
from src.models.unigram_model import UnigramModel

# Init model
model = UnigramModel(3, 5)
# Train model
model.train(X_train, X_dev, y_train, y_dev, iterations=100, init_mode = "gaussian", print_each=10)
# Predictions
mawp_score = model.get_mawp_score()
test_segmentation, test_perplexity = model.predict_segments(X_test)
```

UMM (Unigram Modes Model)

Suppose we have list of string melodies `X_train`, `X_dev`, and `X_test` and their corresponding modes `y_train`, `y_dev`. Then we can get the test segmentation, perplexity, and mawp score this way:

```python
from src.models.unigram_model import UnigramModelModes

# Init model
model = UnigramModelModes(3, 5)
# Train model
model.train(X_train, X_dev, y_train, y_dev, iterations=100, init_mode = "gaussian", print_each=10)
# Predictions
mawp_score = model.get_mawp_score()
test_segmentation, test_perplexity = model.predict_segments(X_test)
```

NHPYLM

Because of the time complexity, we implemented the Nested Hierarchical Pitman-Yor Language Model in Cython, which made the training significantly faster. The implementation of the NHPYLM model is placed in the `src/models/nhpylm` folder called by the NHPYLMModel class implemented in `src/models/nhpylm_model.pyx`. To use the model, we need to set up the Cython code first.

```
python setup.py build_ext --inplace
```

The model could be used in Python this way:

```python
from nhpylm_model import NHPYLMModel

# Init model
model = NHPYLMModel(7, init_d = 0.5, init_theta = 2.0,
                init_a = 6.0, init_b = 0.83333333,
                beta_stops = 1.0, beta_passes = 1.0,
                d_a = 1.0, d_b = 1.0, theta_alpha = 1.0, theta_beta = 1.0)
# Train model
model.train(X_train, X_dev, y_train, y_dev, 200, True, True, print_each_nth_iteration=10)
# Predictions
mawp_score = model.get_mawp_score()
test_segmentation, test_perplexity = model.predict_segments(X_test)
```

NHPYLMModes

NHPYLMModes is the extension of the NHPYLM model based on eight separate NHPYLM submodels for each chant mode. The NHPYLMModes class is also implemented in Cython in the `src/models/nhpylm_model.pyx` file. Therefore, as in the previous case, if we want to use this model, we have to set up the Cython code first.

```
python setup.py build_ext --inplace
```

The model could be used in Python this way:

```python
from nhpylm_model import NHPYLMModesModel

# Init model
model = NHPYLMModesModel(7, init_d = 0.5, init_theta = 2.0,
                init_a = 6.0, init_b = 0.83333333,
                beta_stops = 1.0, beta_passes = 1.0,
                d_a = 1.0, d_b = 1.0, theta_alpha = 1.0, theta_beta = 1.0)
# Train model
model.train(X_train, X_dev, y_train, y_dev, 200, True, True, print_each_nth_iteration=50)
# Predictions
mawp_score = model.get_mawp_score()
test_segmentation, test_perplexity = model.predict_segments(X_test)
```

Note that the NHPYLMModes model needs eight times more time for segment predictions. Be careful with the number of printed iterations.

BERT

We use the official implementation of the Unsupervised Chinese Word Segmentation with BERT Oriented Probing and Transformation paper. We only adapted the code to our task, and we extended the model so that we could pretrain it first. Setting files of our task's BERT model are placed in the `bert/` folder (**config.json**, **vocab.txt**).

For the pretraining process, install the packages with their particular version.

```
pip install transformers==4.27.4 tokenizers==0.13.2
```

Then, move the **bert/config.json** and **bert/vocab.txt** into the root directory. Choose the dataset (**bert/antiphons.zip**, **bert/no4antiphons.zip**, **bert/responsories.zip**), and extract it in the root directory. Then you can pretrain BERT using:

```python
from src.models.bert_model import BERT_Model

model = BERT_Model(working_dir=".")
model.pretrain()
```

The new folder `PretrainedBERT_chant/` should be created. For the training process, change back the version of transformers:

```
pip install transformers==2.8.0
```

And the model could be trained again in the Python code.

```python
from src.models.bert_model import BERT_Model

model = BERT_Model(working_dir=".")
model.train()
```

For the prediction, the dataset format stored in the `datasets/` folder (not **bert/antiphons.zip**, **bert/no4antiphons.zip**, **bert/responsories.zip**) are used.

```python
from src.models.bert_model import BERT_Model

# Init model
model = BERT_Model(working_dir=".")

# Predictions
mawp_score = model.get_mawp_score()
test_segmentation, _ = model.predict_segments(X_test)
```

Random Baseline

The RandomModel generates random segmentation.

```
from src.models.random_model import RandomModel

# Init model
model = RandomModel(1, 7)
# Predictions
mawp_score = model.get_mawp_score()
test_segmentation, test_perplexity = model.predict_segments(X_test)
```

Overlapping N-grams Upper Bound

The Overlapping N-grams is not a segmentation since all segments overlap with others. Furthermore, we can combine more Overlapping N-grams. However, we can measure only the mode classification score. The code sample using the OverlappingNgram class is shown below.

```
from src.utils.loader import prepare_dataset
from src.models.overlapping_ngrams import OverlappigNGrams
from src.eval.pipelines import bacor_pipeline

# Get Data
X, y = prepare_dataset()
# Init models
model1 = OverlappigNGrams(1)
model2 = OverlappigNGrams(2)
model3 = OverlappigNGrams(3)
model4 = OverlappigNGrams(4)
model5 = OverlappigNGrams(5)
model6 = OverlappigNGrams(6)
model7 = OverlappigNGrams(7)
# Train and Fit model
final_segmentation = []
final_segmentation1 = model1.predict_segments(X)
final_segmentation2 = model2.predict_segments(X)
final_segmentation3 = model3.predict_segments(X)
final_segmentation4 = model4.predict_segments(X)
final_segmentation5 = model5.predict_segments(X)
final_segmentation6 = model6.predict_segments(X)
final_segmentation7 = model7.predict_segments(X)
for c1, c2, c3, c4, c5, c6, c7 in zip(final_segmentation1,final_segmentation2,
                                      final_segmentation3,final_segmentation4,
                                      final_segmentation5,final_segmentation6,
                                      final_segmentation7):
  final_segmentation.append(c1+c2+c3+c4+c5+c6+c7)
# Evaluate model
scores, selected_features, trained_model = bacor_pipeline(
    final_segmentation, y, max_features_from_model = 100, include_additative = False, fe_occurence_coef = 10, all_features_vect
```

Evaluation Scores - Usage

Once we have the segmentation (of both training and testing datasets), mawp function, and perplexities, we can evaluate all score functions and visualizations we proposed by a single command.

```
from src.eval.pipelines import evaluation_pipeline

bacor_model = evaluation_pipeline(
    train_segmentation, y_train, test_segmentation, y_test, train_perplexity, test_perplexity, mawp_score,
    max_features_from_model = 100, include_additative = False, fe_occurence_coef=10)
```

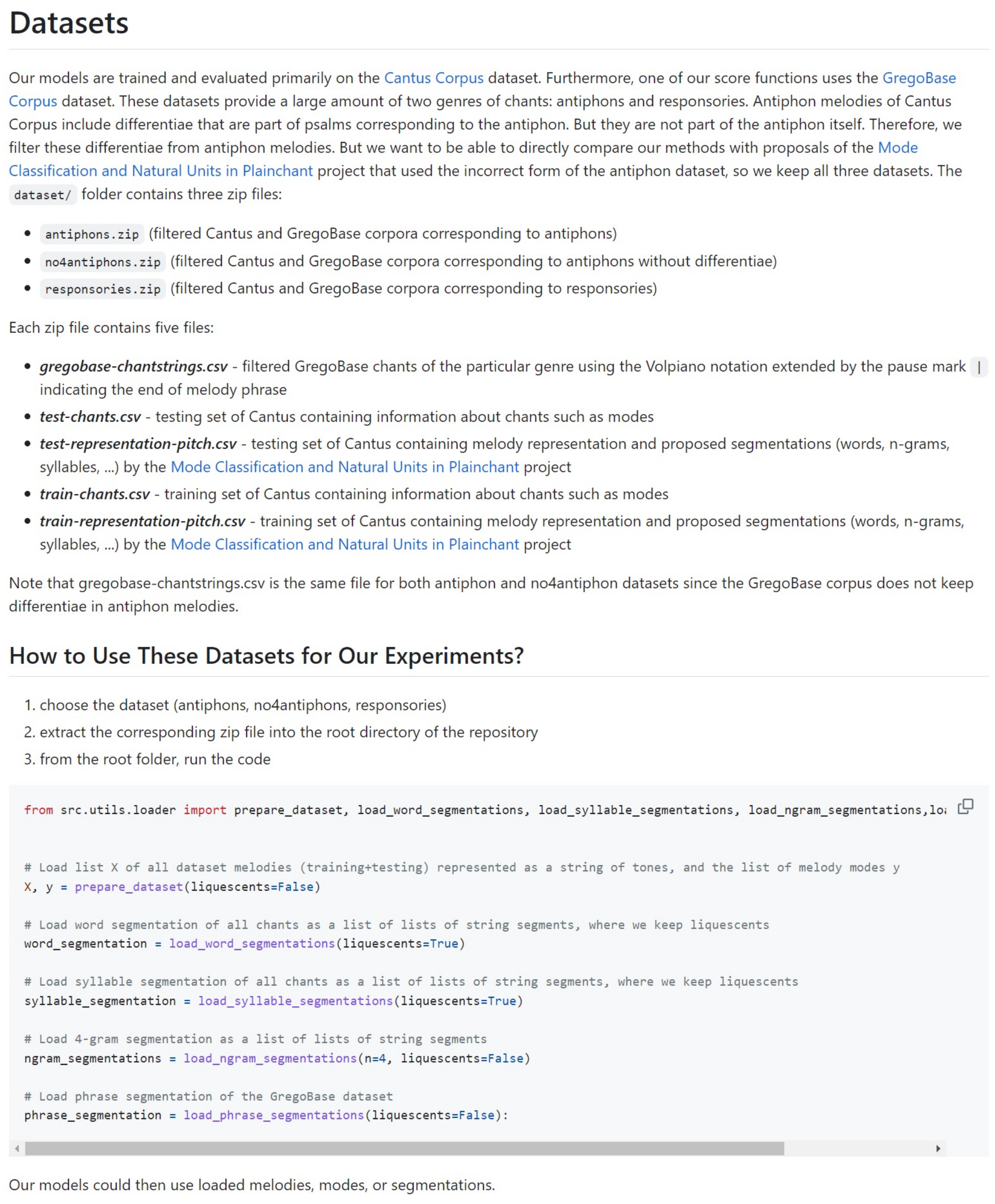
The command will first evaluate and print score results and charts of the training dataset. Then the testing dataset will be evaluated. At the end of the pipeline, the top 100 feature extraction is done and printed with all its related charts.

We measure these score functions

- **perplexity**: the ability of the probability model to predict a sample.
- melody segmentation scores
  - **bacor_accuracy**: mode classification accuracy score of segmentation using the TFIDF vectorizer and SVC classifier
  - **bacor_f1**: mode classification f1 score of segmentation using the TFIDF vectorizer and SVC classifier
  - **nb_accuracy**: mode classification accuracy score of segmentation using the TFIDF vectorizer and Naive Bayes classifier
  - **nb_f1**: mode classification f1 score of segmentation using the TFIDF vectorizer and Naive Bayes classifier
  - **avg_seg_len**: average segment length of all segments in the testing dataset

    - **vocab_size**: the size of segment vocabulary considering the testing dataset - number of unique melodic units used in the segmentation
    - **maww**: how well is melody aligned with words
    - **mawp**: how well is melody aligned with phrases
    - **wtmf**: weighted top mode frequency, weighted (by its frequency) average of percentages that the melodic unit is used in its dominant mode considering all segment occurrences
    - **vocab_levenshtein**: score measuring the diversity of the vocabulary, the average of medians of levensthein distances between segments
    - **wufpc**: weighted unique final pitch count, number of unique last tones of melody segments
- charts
    - **unique segment density**: charts that visualize the percentage of unique segments at the specific position over all segments at that position
    - **average segment length**: charts that visualize average segment length at the specific position of the chant
    - **segment occurrences**: charts that visualize segment occurrences of all vocabulary melodic units and how much modes share the same segments

A.4 datasets.md



Datasets

Our models are trained and evaluated primarily on the Cantus Corpus dataset. Furthermore, one of our score functions uses the GregoBase Corpus dataset. These datasets provide a large amount of two genres of chants: antiphons and responsories. Antiphon melodies of Cantus Corpus include differentiae that are part of psalms corresponding to the antiphon. But they are not part of the antiphon itself. Therefore, we filter these differentiae from antiphon melodies. But we want to be able to directly compare our methods with proposals of the Mode Classification and Natural Units in Plainchant project that used the incorrect form of the antiphon dataset, so we keep all three datasets. The `dataset/` folder contains three zip files:

- `antiphons.zip` (filtered Cantus and GregoBase corpora corresponding to antiphons)
- `no4antiphons.zip` (filtered Cantus and GregoBase corpora corresponding to antiphons without differentiae)
- `responsories.zip` (filtered Cantus and GregoBase corpora corresponding to responsories)

Each zip file contains five files:

- ***gregobase-chantstrings.csv*** - filtered GregoBase chants of the particular genre using the Volpiano notation extended by the pause mark `|` indicating the end of melody phrase
- ***test-chants.csv*** - testing set of Cantus containing information about chants such as modes
- ***test-representation-pitch.csv*** - testing set of Cantus containing melody representation and proposed segmentations (words, n-grams, syllables, ...) by the Mode Classification and Natural Units in Plainchant project
- ***train-chants.csv*** - training set of Cantus containing information about chants such as modes
- ***train-representation-pitch.csv*** - training set of Cantus containing melody representation and proposed segmentations (words, n-grams, syllables, ...) by the Mode Classification and Natural Units in Plainchant project

Note that gregobase-chantstrings.csv is the same file for both antiphon and no4antiphon datasets since the GregoBase corpus does not keep differentiae in antiphon melodies.

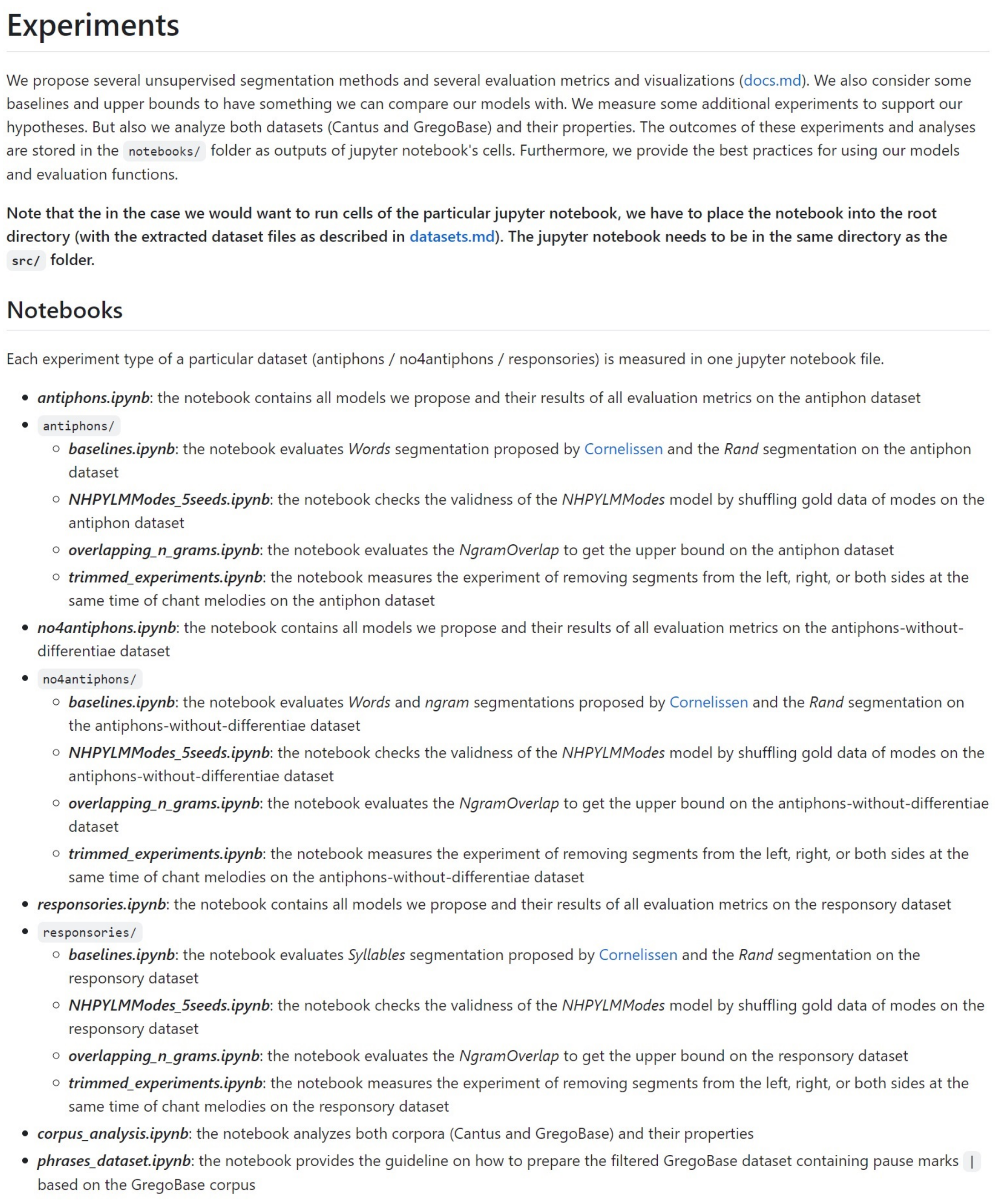
How to Use These Datasets for Our Experiments?

1. choose the dataset (antiphons, no4antiphons, responsories)
2. extract the corresponding zip file into the root directory of the repository
3. from the root folder, run the code

```
from src.utils.loader import prepare_dataset, load_word_segmentations, load_syllable_segmentations, load_ngram_segmentations,lo

# Load list X of all dataset melodies (training+testing) represented as a string of tones, and the list of melody modes y
X, y = prepare_dataset(liquescents=False)

# Load word segmentation of all chants as a list of lists of string segments, where we keep liquescents
word_segmentation = load_word_segmentations(liquescents=True)

# Load syllable segmentation of all chants as a list of lists of string segments, where we keep liquescents
syllable_segmentation = load_syllable_segmentations(liquescents=True)

# Load 4-gram segmentation as a list of lists of string segments
ngram_segmentations = load_ngram_segmentations(n=4, liquescents=False)

# Load phrase segmentation of the GregoBase dataset
phrase_segmentation = load_phrase_segmentations(liquescents=False):
```

Our models could then use loaded melodies, modes, or segmentations.

A.5 experiments.md



Experiments

We propose several unsupervised segmentation methods and several evaluation metrics and visualizations (docs.md). We also consider some baselines and upper bounds to have something we can compare our models with. We measure some additional experiments to support our hypotheses. But also we analyze both datasets (Cantus and GregoBase) and their properties. The outcomes of these experiments and analyses are stored in the `notebooks/` folder as outputs of jupyter notebook's cells. Furthermore, we provide the best practices for using our models and evaluation functions.

Note that the in the case we would want to run cells of the particular jupyter notebook, we have to place the notebook into the root directory (with the extracted dataset files as described in datasets.md). The jupyter notebook needs to be in the same directory as the `src/` folder.

Notebooks

Each experiment type of a particular dataset (antiphons / no4antiphons / responsories) is measured in one jupyter notebook file.

- ***antiphons.ipynb***: the notebook contains all models we propose and their results of all evaluation metrics on the antiphon dataset
- `antiphons/`
  - ***baselines.ipynb***: the notebook evaluates *Words* segmentation proposed by Cornelissen and the *Rand* segmentation on the antiphon dataset
  - ***NHPYLMModes_5seeds.ipynb***: the notebook checks the validness of the *NHPYLMModes* model by shuffling gold data of modes on the antiphon dataset
  - ***overlapping_n_grams.ipynb***: the notebook evaluates the *NgramOverlap* to get the upper bound on the antiphon dataset
  - ***trimmed_experiments.ipynb***: the notebook measures the experiment of removing segments from the left, right, or both sides at the same time of chant melodies on the antiphon dataset
- ***no4antiphons.ipynb***: the notebook contains all models we propose and their results of all evaluation metrics on the antiphons-without-differentiae dataset
- `no4antiphons/`
  - ***baselines.ipynb***: the notebook evaluates *Words* and *ngram* segmentations proposed by Cornelissen and the *Rand* segmentation on the antiphons-without-differentiae dataset
  - ***NHPYLMModes_5seeds.ipynb***: the notebook checks the validness of the *NHPYLMModes* model by shuffling gold data of modes on the antiphons-without-differentiae dataset
  - ***overlapping_n_grams.ipynb***: the notebook evaluates the *NgramOverlap* to get the upper bound on the antiphons-without-differentiae dataset
  - ***trimmed_experiments.ipynb***: the notebook measures the experiment of removing segments from the left, right, or both sides at the same time of chant melodies on the antiphons-without-differentiae dataset
- ***responsories.ipynb***: the notebook contains all models we propose and their results of all evaluation metrics on the responsory dataset
- `responsories/`
  - ***baselines.ipynb***: the notebook evaluates *Syllables* segmentation proposed by Cornelissen and the *Rand* segmentation on the responsory dataset
  - ***NHPYLMModes_5seeds.ipynb***: the notebook checks the validness of the *NHPYLMModes* model by shuffling gold data of modes on the responsory dataset
  - ***overlapping_n_grams.ipynb***: the notebook evaluates the *NgramOverlap* to get the upper bound on the responsory dataset
  - ***trimmed_experiments.ipynb***: the notebook measures the experiment of removing segments from the left, right, or both sides at the same time of chant melodies on the responsory dataset
- ***corpus_analysis.ipynb***: the notebook analyzes both corpora (Cantus and GregoBase) and their properties
- ***phrases_dataset.ipynb***: the notebook provides the guideline on how to prepare the filtered GregoBase dataset containing pause marks `|` based on the GregoBase corpus