Lengoaia eta Sistema Informatikoak Saila

Informatika Fakultatea

Automatic Scansion of Poetry

Manex Aguirrezabal Zabaletak
Iñaki Alegriaren eta Mans Huldenen zuzendaritzapean
Informatikan Doktore titulua eskuratzeko aurkeztutako
TESI-TXOSTENA

Donostia, 2017ko maiatza

Aitari eta amari

...
Edari maitagarria,
tristearen alegria.
Dezu alaitzen begia,
kentzen melankonia,
mutuba ipintzen kantan,
eta errena dantzan.
...

Ardoari jarritako hitz neurtuak — Aita Meagher (1703-1772)

And you know the sun's settin' fast
And just like they say nothing good ever lasts
...

Our Town — Iris DeMent (1961-)

Dediqué mi libro
a una niña de un año,
y le gustó tanto,
que se lo comió.

Gloria Fuertes (1917-1998)

Acknowledgements

I must say I would never finish this work without the help of some people, and I would like to mention them:

- Lehenik eta behin, nire sustengu pertsonal izateagatik, **etxekoak** aipatu nahi ditut: Aita, ama, Endika eta Garazi. Zuek gabe ez bainuke hau egingo, inondik inora ere.

- Eta zer egin inon **lagunik** gabe? Ezin zuek agurtu gabe geratu, Oier eta Ima. Gorka, Beñat, Mikel, zuek ere baduzue meritua nirekin! Oier Lakuntzak ere izan du horrelako tesi informatiko-literario bat sortzearen errua. Milesker!!

- Etxekoak agurtuta, akademiara egingo dut jauzi. Akademian inori eskerrak ematekotan, nire **zuzendari/bidegile** izan ditudanei ematen dizkiet. Iñaki, Mans, Bertol eta Jeff. Zuek gabe benetan, aurkezten dudan lan hau ez litzateke posible. Eta zuek gabe, ez nintzateke gaur naizena izango.

- Orain **pisukideen** ordua. Azken urtea Donostian emanda, nola ahaztuko ditut ba nire pisukideak? Zer egingo nuke ba nik zuek gabe? Bego, Xora, Mendi eta Aintzane! Izugarriak zeate!! But I need to mention a special flatmate, my Best Italian Friend!! Sabrina!! I had so much fun when you were our flatmate!

- Nola ahaztu **bulegokideak**? Nola ahaztu bulegoko krisi komiteak eta gure *tea saloon*-a? Berebiziko garrantzia izan duzue zuek ere, Zuhaitz, Itziar, Arantxa, Olatz, Josu eta Iñigo!! Mila esker!

- Ezin aipatu gabe utzi **iXakideak**! Mundialak zeate! Zer egingo genuke 11etako kaferik gabe? Sanke-rik gabe? mintegiXarik gabe? osteguneko pintxo-poterik gabe?

- Ixa-z aparte **RSAIT** taldea ezin aipatu gabe utzi, haien ateak beti izan baitituzte zabalik edozer behar izanez gero laguntzeko. Nire atea ere beti izango duzue zabalik, Elena, Basi, Igor ta konpainia. Robotika taldean aipamen berezia, **Aitzol Astigarragari** egin nahi diot. Nork esango zuen, 2010 inguruan Bertsotarako Arbel Digitala garatzera animatu zen mutila bertso sorkuntza automatikoko munduan sartuko zela? Nola ahaztu, Aitzol, eman didazun laguntza osoa? Zuk ere nire ateak beti zabalik izango dituzu. :-)

- Ezin ahaztu **Informatika Fakultate osoa**, lankide guztiek etxean bezala sentituarazi bainaute. Itziar Irigoienen laguntza gauza matematiko-estatistikoetan, Elena eta Asunekin sortutako elkarrizketa espontaneoak, ...

- Kontutan izanda poesia eta bertsoekin egin dugula lan azken urteotan, **Bertsozale Elkarteari** ere eskerrak bidali nahi dizkiot, bereziki ikerkuntza sailari eta bere arduradun Karlos Aizpuruari bere babesagatik. Eta ezin ahaztu Nere Erkiaga, Mintzolara joaten naizen bakoitzean etxean bezala sentituarazten nau eta. Eskerrik asko zuri ere, Nere!

- Before finishing, I must acknowledge the **University of Delaware** and especially, the **Department of Linguistics and Cognitive Sciences** for receiving me as a visiting student in 2014. I still remember the so interesting classes that we got; CompLing, Phonology-1, Machine Learning (at the Computer and Information Sciences Dept.)... I must thank Jeff and also Vijay Shanker for allowing me to learn at their classes.

- But, what's a **department** without **people**? Thanks a lot to the people that I met at the Dept,: David, Taylor, Amanda, Gordon, Adam, Justin, Curt, Lan, Yugyeong, Hyunjin, Young-eun (thanks for being my Korean teacher), Claudia, Rebecca, ... I especially want to mention to one of them, for making me so happy at that time.

- I also want to thank all the **friends** that I met in **Delaware**, especially: Samuel, Ansgar, Tim, Zulia, Juliet, Karla, Ryan, Carlos, Dina, Alex ...

Baina, hau buketu aurretik, eskertza berezi bat egin nahiko nuke. Oso gaztea nintzela, 10 urte inguru nituela, pertsona batek iada "*doctor*" izenez deitzen ninduen. Pasa zen denbora, eta informatika ikasketak bukatu nituenean, Karrera Bukaerako Proiektuaren defentsan egon zen hau, aitarekin,

amarekin eta izeba Maribelekin batera. 10 hilabete ondoren, Master amaierako lana defendatu nuen, eta arazo batzuk medio, berak ezin izan zuen etorri. Nork esango zuen, hortik hiru hilabetetara joango zitzaigula? Nork esango zuen tesi honen defentsan ezingo zukeela egon? Patuak hala erabaki zuen, ordea. Horregatik, aipamen berezi bat egin nahi diot nire osaba zenari, **Jose Ramon Insaustiri**. Mila esker denagatik, Joserra.

Contents

List of Figures

List of Tables

CHAPTER I

Introduction

I.1 Motivation

O Captain! my Captain! our fearful trip is done,
The ship has weather'd every rack, the prize we sought is won,
The port is near, the bells I hear, the people all exulting,
While follow eyes the steady keel, the vessel grim and daring;
But O heart! heart! heart!
O the bleeding drops of red,
Where on the deck my Captain lies,
Fallen cold and dead.

...

My Captain does not answer, his lips are pale and still,
My father does not feel my arm, he has no pulse nor will,
The ship is anchor'd safe and sound, its voyage closed and done,
From fearful trip the victor ship comes in with object won;
Exult O shores, and ring O bells!
But I with mournful tread,
Walk the deck my Captain lies,
Fallen cold and dead.

The above poem is entitled *Oh captain! My captain!*, and is one of the most well-known poems written by American poet Walt Whitman in honor of Abraham Lincoln. In this poem there are some moments in which rhythm plays an important role, such as "*The port is near, the bells I hear, ...*", or

also "*our fearful trip is done*" and "*his lips are pale and still*". This rhythm, created by repetitive patterns, can evoke specific emotion in people. It can make the difference between a regular discourse or a memorable discourse. Consider Barack Obama's speech following his victory in the Iowa caucus in 2008. The recurrent *DEH-DUM-DEH-DUM-...* rhythm heard in the speech is by no means fortuitous.

They said this day would never come.
They said our sights were set too high.
...

The slogans "Yes We Can" and "Change We Need" used by Barack Obama are not so memorable purely by chance, neither is "Free at Last", recited by Martin Luther King.

In the above example by Walt Whitman, the rhythm of the stanza is repeated throughout the three stanzas. But usually, each line's sound sequence is repeated across lines. This pattern of recurrence is called **meter**. Let us consider a stanza from Lewis Carroll's Jabberwocky poem [Carroll, 1982]:

One, two! One, two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.

Scansion involves marking the rhythmic structure of a poem, marking the units that are emphasized when the poem is read aloud. Those emphasized sound units appear recurrently, such as, the DEH-DUM pattern from "he **left** it **dead**" or the ending of some lines (*snack* and *back*). If we **scan** this poem, then, we will have encoded all that information.

One, ***two!***‖ *One,* ***two!***‖ *And* ***through***‖ *and* ***through***
The ***vor***‖*pal* ***blade***‖ *went* ***snick***‖*er-****snack****!*
He ***left***‖ *it* ***dead****,*‖ *and with*‖ *its* ***head***
He ***went***‖ *ga****lum***‖*phing* ***back****.*

The syllables that are marked with bold font in the lines (*two*, *through*, *vor*, *blade*, ...) are **stressed** and the others are **unstressed**. As previously mentioned, there is a repetitive sequence of DEH-DUM, or unstressed-stressed, sounds. This grouping is called a **foot**. The repetition of a sound at the end of some lines is called **rhyme**. Although scanning a line of poetry

involves all these characteristics, in this work I focus mainly on the rhythm found in the syllable stress patterns.

As rhythm affects readers of poetry, so it plays a key role in the experience of listening to music. For example, if we listen the song Leroy Anderson's "*The Typewriter*" and Antonin Dvorak's "*New World Symphony's Largo*", feelings are very different. In music, rhythm is one of the aspects that contribute to the emotions to be transmitted, together with pitch, timbre, texture and so on. In a similar way that different kinds of music create different feelings, so too do different poems. Of course, poets have other poetic devices to convey meaning, such as alliteration, rhymes, repetition of syntactic elements, among others.

But, what is the value of poetry scansion? Imagine that we have to decide whether the author of a poem is Shakespeare or Dr. Seuss.

Then I went for the eggs of a Long-Legger Kwong.
Now this Kwong . . . well, she's built just a little bit wrong,
For her legs are so terribly, terribly long
That she has to lay eggs twenty feet in the air
And they drop, with a plop, to the ground from up there!
So unless you can catch 'em before the eggs crash
You haven't got eggs. You've got Long-Legger hash.

If we scan this poem for its rhythmic structure, we will easily realize the high number of DEH-DEH-DUM sound patterns ("*Then I went for the eggs...*", "*And they drop, with a plop, to the ground...*"). By knowing this, we could easily discard Shakespeare as the author of the poem as Shakespeare wrote his poetry generally using iambic meter (DEH-DUM sound pattern). In fact, this excerpt is from Dr. Seuss' book *Scrambled Eggs Super!* [Seuss, 1953].

In this work I try to automatically assign each of those "DEH" or "DUM" sounds to each syllable. However, which pieces of information are relevant in order to assign stresses to syllables in English poems? This is an important question I aim to answer. Going further, if the elements of the language that describe stresses in English are known, is it possible to scan (analyze) poems in other languages using such elements?

For example, let us consider this excerpt from the Sonnet XXIII by Garcilaso de la Vega:

En tanto que de rosa y azucena

se muestra la color en vuestro gesto,
y que vuestro mirar ardiente, honesto,
enciende al corazón y lo refrena.

If we were to mark the stresses in the poem, we would mark them in this way:

En ***tan****to que de* ***ro****sa y azu****ce****na*
se ***mues****tra la co****lor*** *en vuestro* ***ges****to,*
*y que vuestro mi****rar*** *ar****dien****te, ho****nes****to,*
*en****cien****de al cora****zón*** *y lo re****fre****na.*

Following a notation typically used in poetry scansion, we would mark stressed syllables with a slash symbol (`/`) and the unstressed ones with the letter `x`:

En	*tan*	*to*	*que*	*de*	*ro*	*sa_y*	*a*	*zu*	*ce*	*na*
x	/	x	x	x	/	x	x	x	/	x
se	*mues*	*tra*	*la*	*co*	*lor*	*de*	*vues*	*tro*	*ges*	*to*
x	/	x	x	x	/	x	x	x	/	x
y	*que*	*vues*	*tro*	*mi*	*rar*	*ar*	*dien*	*te_ho*	*nes*	*to*
x	x	x	x	x	/	x	/	x	/	x
en	*cien*	*de_al*	*co*	*ra*	*zón*	*y*	*lo*	*re*	*fre*	*na*
x	/	x	x	x	/	x	x	x	/	x

I.2 From marking stresses to finding structure in raw text

Labeling sentences is a common task in Natural Language Processing (NLP). This task can be as simple as marking the part-of-speech tags in a sentence.

pronoun	noun	verb	adjective
My	*dog*	*is*	*brown*

And, it can be made harder by marking the whole groupings from a sentence, such as, "My dog" as a noun phrase. This leads to a richer analysis, for instance:

noun-phrase		verb	preposition	location
My	*friend*	*is*	*from*	*Eastwood*

Yet, this analysis may not be quite so simple. How do we know that Eastwood is referring to a location, and not to a surname? We can deduce this information by looking at the surrounding elements, or the context ("My friend is from ...").

Poetic scansion can be done in a similar way, using computational techniques. As previously, the context is helpful for scansion too. For example, consider the first two lines from *Paradise Lost*, by John Milton:

No more of talk where God or Angel Guest
With Man, as with his friend, familiar us'd

The even syllables of the first line —*more*, *talk*, *God*, *An-* and *guest*—for most readers tend to appear naturally prominent. These beats usually correspond to nouns, verbs, adjectives and adverbs. The problem arises when tagging the second line:

No	more	of	talk	where	God	or	An-	gel	Guest
x	/	x	/	x	/	x	/	x	/

With	man,	as	with	his	friend	fa	mi	liar	us'd
x	/	x	?	x	/	x	/	x	/

In the second line, the second occurrence of the word *with* does not sound like a stressed word, but by looking at the context (previous and next stresses, previous line) it seems to be a stressed syllable, by inertia. Identifying the important aspects that contribute to English poetic scansion—words natural prominence, POS-tag, among others, is one of the goals of this thesis.

If a principled way of marking the prominences in English poems can be defined, can this be applied to other languages, such as Spanish? Consider this excerpt from the poem "Fue una clara tarde, triste y soñolienta" by Antonio Machado:

Adiós para siempre la fuente sonora,
del parque dormido eterna cantora.
Adiós para siempre; tu monotonía,
fuente, es más amarga que la pena mía.

Instinctively, the first two lines have a clear DEH-DUM-DEH sound pattern,

*A**diós** para **siem**pre la **fuen**te so**no**ra,*
*del **par**que dor**mi**do e**ter**na can**to**ra.*

but this is interrupted in the third line ("A**diós** para **siem**pre; tu monoto**ní**a"). This poses an intriguing question; should we mark the stresses as "**mo**noto**ni**a"?

Finally, without resorting to linguistic information nor previous annotations, is it possible to tag a poem? Can we find structure in raw text? Consider extracts of these two poems, the first from "Ash wednesday" by T.S. Elliot and the second from "Ni naiz" by Xabier Lete (in Basque):

...

Because I do not hope to know again
The infirm glory of the positive hour
Because I do not think
Because I know I shall not know
The one veritable transitory power
Because I cannot drink
There, where trees flower, and springs flow, for there is nothing again

...

...

Ni naiz
erreka zikinen iturri garbiak
aurkitu nahi dituen poeta tristea.
Ni naiz
kaleetan zehar neguko eguzkitan
lanera dijoan gizon bakartia.
Ni naiz
lorerik gabe gelditzen ari den
ardaska legorra,
ni naiz
pasio zahar guztiak kixkali nahi dituen
bihotz iheskorra.

...

In the first poem, similar structures such as "*Because I do not hope*", "*Because I do not think*" and "*Because I cannot drink*" are repeated throughout the poem. In the second one, "*Ni naiz*" is continuously repeated, structuring the whole poem, and sounds are analogous in "*po**e**ta **tris**te**a***", "*gi**zon** ba**kar**ti**a***", "*ar**das**ka **le**go**rra***" and "*bi**hotz** i**hes**ko**rra***".

With this goal in mind, I first analyze poems in English using a rule-based system. After that, some statistical methods that rely on tagged data

have been developed. The interest in empirical methods arises from the fact that in order to create these models we only need tagged data, and not linguistic knowledge. Using the same methodology and information, I study how empirical systems can be extrapolated to any language so as to determine their applicability. Finally, I have also performed various experiments using what is called *unsupervised learning*, which learns directly from data without any tagged information. As tagged data is not necessary, these last models are the most affordable options, and also easily applicable to more languages.

At the beginning of my investigation [Agirrezabal, 2012], our goal was to create Natural Language Generation systems that would be able to automatically generate Basque poems, as in Manurung [2004], Gervás et al. [2005], Greene et al. [2010], Oliveira [2012], Hartlová et al. [2013], Toivanen et al. [2013], Gervás [2014]. We were able to create some approaches for generation [Agirrezabal, 2012, Agirrezabal et al., 2013b, Astigarraga et al., 2013, 2014], and then we created some of the modules required to assemble a Natural Language Generation tool, including Content Determination, choosing the best ending for a verse,... Due to its complexity, however, it was not sufficient to fully produce text. I think that a better understanding of poetry, for instance in terms of meter, will allow us to create more reliable systems to analyze and also to generate verses. Apart from this, if the evaluation of computer-generated natural language is challenging, the problem becomes even more challenging when evaluating computer-generated poems [Lamb et al., 2016], as evaluation can be (almost) completely subjective. Recent efforts in evaluation of poetry-generation toolkits using microblogging services show that a more objective evaluation is possible [Veale, 2015].

I.3 Research questions

In this dissertation I try to answer some research questions, which I expect to be relevant to the research community.

1. Which are the informative features when analyzing a poem and how can we capture them?

2. Does language-specific linguistic knowledge contribute when analyzing poetry?

3. Is it possible to analyze a poem without having information about language? Is such analysis something that can be learnt?

I.4 Tasks

In order to answer the above research questions, I worked following this methodology and performing the following tasks:

- With English poetry scansion as a basis, develop an automatic poetry scansion system for the English language, based on linguistic rules.
- Collect a corpus of written and scanned poetry in English to test the scansion system.
- Training of data-based models using several computational tools, based on the collected data and extended language-specific features. Use simple features and extended language-specific features.
- Collection of additional corpora in other languages and annotation when necessary.
- Extrapolation of previous data-driven models to other languages.
- Try to infer poetic stress patterns directly from data without using tagged information.

I.5 Structure of the thesis

The structure of the thesis is as follows: In Chapter II I first discuss, in general, the problem of scansion and then go on to discuss the tradition of three different languages (subsections II.1.2, II.1.3 and II.1.4). Then, in section II.2, some previous work on automatic analysis of poetry are shown and following this some typical approaches to sequence modeling are reviewed. Chapter III covers the different corpora employed for the training and evaluation of different models and also addresses how poetic information is encoded in these corpora. In Chapter IV I explore the algorithms used for scansion. This chapter is divided in three main sections: rule-based scansion (section IV.1), supervised learning (section IV.2) and unsupervised learning (section IV.3), together with a final discussion about the presented methods. In chapter V the performed experiments are shown together with their actual performance in previously mentioned data and in chapter VI I close the dissertation with a discussion, answering the research questions and proposing some possible future directions.

I.6 Publications

I.6.1 Directly related to the dissertation

1. Agirrezabal, M., Alegria, I., and Hulden, M. (2017, May). Poesiaren eskantsio automatikoa: bi hizkuntzen azterketa. In *IkerGazte*, volume 1, UEU.

2. Agirrezabal, M., Alegria, I., & Hulden, M. (2016, December). Machine Learning for the Metrical Analysis of English Poetry. *International Conference on Computational Linguistics (COLING 2016)*, pp.:772-781

3. Agirrezabal, M., Astigarraga, A., Arrieta, B., & Hulden, M. (2016). ZeuScansion: a tool for scansion of English poetry. *Journal of Language Modelling*, 4(1), 3-28.

4. Agirrezabal, M., Heinz, J., Hulden, M., & Arrieta, B. (2014, January). Assigning stress to out-of-vocabulary words: three approaches. *In Proceedings on the International Conference on Artificial Intelligence (ICAI)* (p. 1). *The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).*

5. Agirrezabal, M., Arrieta, B., Astigarraga, A., and Hulden, M. (2014). 1986-2013 arteko bertsolari txapelketa nagusien analisi estatistikoa. *Bertsolari Txapelketa Nagusia.*

6. Agirrezabal, M., Arrieta, B., Astigarraga, A., & Hulden, M. (2013, August). POS-tag based poetry generation with WordNet. *In Proceedings of the 14th European Workshop on Natural Language Generation* (pp. 162-166).

7. Agirrezabal, M., Arrieta, B., Astigarraga, A., and Hulden, M. (2013c). ZeuScansion: a tool for scansion of English poetry. *In the Finite State Methods and Natural Language Processing Conference*, page 18.

8. Agirrezabal, M., Arrieta, B., Astigarraga, A., and Hulden, M. (2013a). Bota bertsoa, eta guk aztertuko dugu: azken urteetako bertsolari txapelketa nagusien analisia. *Elhuyar: zientzia eta teknika*, (300):46–49.

9. Agirrezabal, M., Alegria, I., Arrieta, B., & Hulden, M. (2012, July). Finite-State Technology in a Verse-Making Tool. *In the Finite State Methods and Natural Language Processing Conference* (pp. 35-39).

10. Agirrezabal, M., Alegria, I., Arrieta, B., & Hulden, M. (2012, April). BAD: An assistant tool for making verses in Basque. *In Proceedings of the 6th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities* (pp. 13-17). *Association for Computational Linguistics.*

I.6.2 Collaborations

1. Gamallo, P., Pichel, J. R., Alegria, I., & Agirrezabal, M. (2016). Comparing two Basic Methods for Discriminating Between Similar Languages and Varieties. VarDial 3, 170.

2. Agirrezabal, M., Gonzalez-Dios, I., and Lopez-Gazpio, I. (2015). Euskararen sorkuntza automatikoa: lehen urratsak. In *IkerGazte*, volume 1, pages 15-23. UEU.

3. Astigarraga, A., Jauregi, E., Lazkano, E., and Agirrezabal, M. (2014). Textual coherence in a verse-maker robot. *In Human-Computer Systems Interaction: Backgrounds and Applications 3,* pages 275–287. *Springer International Publishing.*

4. Osinalde, M., Astigarraga, A., Rodriguez, I., & Agirrezabal, M. (2013). Towards Basque Oral Poetry Analysis: A Machine Learning Approach. *In Student Workshop at Recent Advances in Natural Language Processing (RANLP)* (pp. 119-125).

5. Perez-de-Vinaspre, O., Oronoz, M., Agirrezabal, M., & Lersundi, M. (2013, July). A Finite-State Approach to Translate SNOMED CT Terms into Basque Using Medical Prefixes and Suffixes. *In the Finite State Methods and Natural Language Processing Conference* (pp. 99-103).

6. Astigarraga, A., Agirrezabal, M., Lazkano, E., Jauregi, E., & Sierra, B. (2013, June). Bertsobot: the first minstrel robot. *In Human System Interaction (HSI), 2013 The 6th International Conference* on (pp. 129-136). *IEEE.*

7. Agirrezabal, M., Alegria, I., and Hulden, M. (2012c). Using foma for language-based games. *1st Workshop on Games and NLP (GAMNLP-12), a special session at JapTAL 2012 (the 8th International Conference on Natural Language Processing), Kanazawa, Japan.*

CHAPTER II

Scansion and Sequence labelling

II.1 Scansion

Scansion is a well-established form of poetry analysis which involves marking the prosodic meter of lines of verse and possibly also dividing the lines into feet. The specific technique and scansion notation may differ from language to language because of phonological and prosodic differences, and also because of different traditions regarding meter and form. Scansion is traditionally done manually by students and scholars of poetry.

There are different metrical systems within the wide range of poetic traditions around the world. These metrical systems are:

- Quantitative meter: This is used in Greek and Latin and the patterning corresponds to the length of the syllable.
- Syllabic meter: The number of syllables used in each line governs the meter. This system is used in French and Japanese.
- Accentual meter: This was the most popular meter in Old English, and it is still used in contemporary poetry. The line pattern is guided by a regular number of stressed syllables, regardless of the number of unstressed syllables.
- Accentual-syllabic meter: This is by far the most used meter in English poetry, where the pattern states the number of syllables that will be in each line and also the ordering of unstressed-stressed syllables, allowing some variations.

The following examples illustrate these metrical systems. An example of a poem in quantitative meter is Virgil's *The Aeneid: Book 1*, and starts as:

Arma virumque canō, Trōiae quī prīmus ab orīs
Ītaliam, fātō profugus, Lāvīniaque vēnit
lītora, multum ille et terrīs iactātus et altō
...

Each of these lines follows a continuous pattern of dactylic syllable groups[1] with some possible variation. The poem sounds like "*Aaarma viruuumque ca...*". In the case of poems in syllabic meter, a common example is the French poem "*Les Fleurs du mal*", by Charles Baudelaire:

Si le viol, le poison, le poignard, l'incendie,
N'ont pas encore brodé de leurs plaisants dessins,
Le canevas banal de nos piteux destins,
C'est que notre âme, hélas ! n'est pas assez hardie.

These lines are alexandrines, and as such, must contain 12 syllables. 68% of the lines of "Les Fleurs du mal" are alexandrines and around 13% octosyllabic [Coates, 1998]. The remaining lines contain a different number of syllables. As it may be mentioned later in this section, Old English poetry was written in accentual meter. An example of accentual verse written in Old English is the epic poem "*Beowulf*", whose excerpt is shown below:

wuldres wealdend, woroldare forgeaf;
Beowulf wæs breme (blæd wide sprang),
Scyldes eafera Scedelandum in.
Swa sceal geong guma go de gewyrcean,

The lines in *Beowulf* follow a constant pattern of four stresses and a caesura between two of those stresses. The last metrical system, the accentual-syllabic, is the one followed by most poets in English, such as William Shakespeare, Cristopher Marlowe and so on. Below a verse from "*Paradise Lost*" by John Milton can be observed:

No more of talk where God or Angel Guest
With Man, as with his Friend, familiar us'd
To sit indulgent, and with him partake
Rural repast, permitting him the while

In the next sections, more thorough analyses of these accentual-syllabic poems will be given, as the main experiments have been done in English.

[1]Dactyl: A pattern with a long syllable followed by two short syllables.

II.1.1 Notation in scansion

When metrically analyzing poetry, syllable prominence can be marked using a two-level marking, either stressed or unstressed. Some researchers, however, use a higher level marking, probably to quantify the level of prominence in each syllable. Some examples in which different notation have been used are provided in these lines.

A two-level notation has been used in some works, such as Fussell [1965], Steele [1999] and the Princeton Encyclopedia of Poetry and Poetics [Preminger et al., 2015]. Other authors use a three-level notation, or even a four-level notation. Such works include Jespersen [1933], Corn [1997], Hayes et al. [2012].

In the rule-based method a three-level notation is used internally, which later is reduced to two levels. In data-driven methods, however, only two-levels are used. In this dissertation, stressed syllables are marked with a slash symbol (`/`) and unstressed ones with the `x` character, and when secondary stresses are needed the backslash symbol (\) is used. In the corpora (presented in chapter III), stressed syllables and unstressed ones are marked with the `+` and `-` symbols, respectively. All the corpora used in this work are annotated with two levels.

In the following section meter in English, Spanish and Basque poetic traditions is discussed. Some theoretical concerns about each language's poetic tradition are introduced and their typical metrical structures are shown, along with some examples.

II.1.2 Tradition in English

In this subsection English poetic tradition is discussed. In this work, when I speak about English poetry, I do not refer only to poetry from the United Kingdom, but to all poetry in English. There are several books that provide a general introduction to prosody in English poetry, for example, Corn [1997] or Steele [1999].

Different approaches for the rhythmi-metrical analysis of poetry have been proposed [Küper, 2012], such as traditional metrics [Fussell, 1965, Fry, 2010], generative metrics [Halle and Keyser, 1971, Fabb and Halle, 2008] (metrical grid theory), the Russian formalist school [Tarlinskaja, 2006], cognitive metrics [Tsur, 2008, Lilja et al., 2012], and the optimality-theory approach [Hanson and Kiparsky, 1996].

The metrical system used in English poetry was accentual until the writings of Chaucer, considered the father of English literature. Classical exam-

ples of accentual poetry in Old English include *Beowulf* and *Piers Plowman*. After that until the 19th century English poetry was accentual-syllabic. Gerard Manley Hopkins (with his sprung rhythm), Walt Whitman and Amy Lowell among others, started to reduce the dominance of accentual-syllabic verse. Nowadays, accentual poetry is present in nursery rhymes, country poetry and rap [Gioia, 2003].

Poems in English verse have repeating patterns of syllable stresses, better known as feet. According to the number of stresses each of these feet has, two meter groups can be found: duple and triple meters. Duple metered feet contain two syllables and triple metered feet will have three syllables. Different metrical patterns are used in poetry, and below are listed some of the most common ones in English [Baldick, 2015]:

- Iambic meter: This is a duple meter, and by far the most common meter in modern English poetry, such as in Shakespeare's sonnets [Shakespeare, 1609]. An example of this meter would be the word "bal-**loon**".

- Trochaic meter: Another duple meter, not as common as iambic meter, but found in poems such as *The Song of Hiawatha* by Henry W. Longfellow [Longfellow, 1855] or *The Tyger* by William Blake[2] [Blake and Lincoln, 1994]. The word "**jun**-gle" is trochaic.

- Dactylic meter: A triple meter, found in poems such as *The Voice* by Thomas Hardy [Monroe, 1917] and in *The Gashlycrumb Tinies* by Edward Gorey [Gorey, 1963]. The word "**ac**-ci-dent" follows this stress-pattern.

- Anapestic meter: A common triple meter found in humorous or comic poems, for example, "but I'm **tel**-ling you **Liz**" from *Scrambled Eggs Super* by Dr. Seuss [Seuss, 1953] or in traditional Irish limericks.

In the following table, an example of each meter can be seen. The stresses are marked in bold. The iambic example is from Cristopher Marlowe's poem "*The Passionate Shepherd to His Love*", the anapestic one is from Dr Seuss' "*Scrambled Eggs Super!*", the trochaic poem is an excerpt from "*The Song of Hiawatha*", by Henry Wadsworth Longfellow, and the dactylic one from "*The Voice*", by Thomas Hardy.

[2]This is a trochaic poem with catalectic lines. But it could also be considered an iambic poem with acephalous lines.

Iambic meter	Anapestic meter
Come ***live*** *with* ***me*** *and* ***be*** *my* ***love,***	*and I* ***don't*** *like to* ***brag,*** *but I'm* ***tell****ing you* ***Liz***
And ***we*** *will* ***all*** *the* ***plea****sures* ***prove,***	*that* ***speak****ing of* ***cooks*** *I'm the* ***best*** *that there* ***is***
That ***Val****leys,* ***groo****ves,* ***hills*** *and* ***fields,***	*why* ***on****ly last* ***Tues****day when* ***mo****ther was* ***out***
Woods, *or* ***steep****y* ***moun****tain* ***yields.***	*I* ***real****ly cooked* ***some****thing worth* ***talk****ing a****bout***
Trochaic meter	Dactylic meter
Can *it* ***be*** *the* ***sun*** *desc****en****ding*	**Wo**man much **missed**, how you **call** to me, **call** to me,
O'er *the* ***le****vel* ***plain*** *of* ***wa****ter?*	**Say**ing that **now** you are **not** as you **were**
Or *the* ***Red*** *Swan* ***float****ing,* ***fly****ing,*	**When** you had **changed** from the **one** who was **all** to me,
Wound*ed* ***by*** *the* ***ma****gic* ***ar****row,*	**But** as at **first**, when our **day** was **fair**.

The length of a metrical line is expressed based on the number of feet found in the verse, thus a dimeter has two feet, a trimeter three, a tetrameter four, and so on (pentameter, hexameter, heptameter,...).

One of the most used meter in English is iambic pentameter. Let us consider an excerpt from the *Sonnet no. 18* by William Shakespeare:

> *Shall I compare thee to a summer's day?*
> *Thou art more lovely and more temperate:*
> *Rough winds do shake the darling buds of May,*
> *And summer's lease hath all too short a date:*
> ...

An English speaker can realize in this excerpt from a Shakespeare sonnet that there is a recurrent pattern of two syllables. Each line, as it is a pentameter, is composed of five iambic feet. Other metrical patterns, such as trochaic, anapestic or dactylic, are not as common as iambic meter.

Metrical variation

In the previous lines, the presented example written in iambic pentameter which was quite clear. How a particular line of verse *should* be scanned, however, is often a matter of contention. Consider a line from the poem *Le Monocle de Mon Oncle* by Wallace Stevens [1923]:

I wish that I might be a thinking stone

Here, things are far less clear. This line can, for example, be analyzed as five iambic feet, or as one iamb, followed by a pyrrhic foot,[3] followed by two stressed syllables, followed by two more iambs. The issue here is that both analyses must be given as correct, which contrasts with typical NLP labeling tasks, in which usually only one analysis is defined as correct. The following represents several analyses of the line in question.

[3]Pyrrhic foot: Two unstressed syllables [xx].

	I	*wish*	*that*	*I*	*might*	*be*	*a*	*think*	*ing*	*stone*
(1)	x	/	x	/	x	/	x	/	x	/
(2)	x	/	x	x	/	/	x	/	x	/
(3)	x	/	x	/	/	/	x	/	x	/
(4)	x	/	x	x	x	/	x	/	x	/

The first variant is the meter most likely intended by the author. The second line represents the above mentioned alternative scansion. The third and fourth lines show the output of the software tools Scandroid [Hartman, 2005] and ZeuScansion [Agirrezabal et al., 2016b], respectively.

Sometimes a line's analysis can be different from the expected one. In fact, well-known poems usually include some metrical variation; this is a stylistic device to break monotony and provide elements of surprise and variation to the reader. In the poem *The More Loving One* by W. H. Auden [Auden, 1960], the poet varies the meter several times. An interesting case in point is the following stanza

*Ad**mi**rer **as** I **think** I **am***
*of **stars** that **do** not **give** a **damn**,*
*I **can**not, **now** I **see** them, **say***
*I **missed** one **ter**ribly **all day***

where the natural flow of the last line is scanned as two iambs and a double iamb.[4] While the poem itself is written in iambic tetrameters, this last line illustrates the author assigning extra emphasis on the final part: *all day*.

The challenges of scansion

Scansion is, then, the analysis of rhythmic structure in verse. But what makes it difficult? In the following, I discuss some of the immediate obstacles that have to be overcome to provide accurate annotations of rhythm and stress:

(a) Lexical stress patterns do not always apply

(b) Dividing the stress pattern into feet

(c) Dealing with out-of-vocabulary words

[4]Double iamb: two unstressed syllables and two stressed syllables [xx//].

(a) Lexical stress patterns do not always apply

The primary piece of information necessary for performing metrical scansion is the *lexical stress* of words. While other elements are also important, the inherent lexical stress of a word is indispensable for the task. Consider the first line of Thomas Hardy's *The voice* [Monroe, 1917, p. 131]:

Woman much missed, how you call to me, call to me

If we were to simply perform scansion by marking the primary (/), secondary (\), and unstressed syllables (x) along the line as provided for the individual words in a dictionary, the result would be

wo	*man*	*much*	*missed*	*how*	*you*	*call*	*to*	*me*	*call*	*to*	*me*
/	x	/	\	/	/	/	x	/	/	x	/

which is not the pattern we are looking for. This poem is in fact composed of four quatrains[5], where each line is written in dactylic tetrameter throughout, which leads to the following analysis for this line.

wo	*man*	*much*	*missed*	*how*	*you*	*call*	*to*	*me*	*call*	*to*	*me*
/	x	x	/	x	x	/	x	x	/	x	x

As is obvious, we have to know the *prosodic stress* of the line in order to calculate the meter of the poem, but simply knowing the lexical stress of each of the words will not suffice. The lexical stress is the relative emphasis inherent to certain syllables in a word, independently of the word's context. The prosodic stress shows the prominence of each of the syllables within a sentence.

(b) Dividing the stress pattern into feet

The prosodic stress location is important, but knowledge of it is still not sufficient to obtain the intended overall meter of a poem. In order to analyze the meter, each line needs to be divided into plausible feet. Returning to the above example by Thomas Hardy (presented in section a), it should be determined that the poem's lines are composed of four dactyls, and thus, that its meter is dactylic tetrameter.

[5]Quatrain: a stanza containing four lines.

(c) Dealing with out-of-vocabulary words

Automatic scansion is made considerably more difficult by the presence of out-of-vocabulary words. Although the lexical stress of words is not sufficient for scanning a line of poetry, it can be a necessary element. For some words, however, it is not available in standard dictionaries. Let us suppose that we are scanning the following line from Henry Wadsworth Longfellow's poem "*The song of Hiawatha*" [Longfellow, 1855, p. 39]:

By the shores of Gitche gumee

Here, most dictionaries would lack entries for either *Gitche* or *gumee*. For such cases, an informed method or algorithm is needed for assigning lexical stress to out-of-vocabulary words. The use of rare, made-up, or unknown words is, of course, common in poetry. They appear as a result of atypical spellings, are derived through complex morphological processes, or are just nonsensical words coined for the occasion (cf. John Lennon's *The faulty Bagnose* or *Jabberwocky* by Lewis Carroll, 1982). Usually, the character names in poems also do not appear in dictionaries, and so their scansion cannot be inferred from such knowledge sources. This problem is exacerbated in older poetry (e.g., *Beowulf*). Failure to correctly indicate primary stress in such unknown words results in a lower accuracy of automatic scansion systems.

II.1.3 Tradition in Spanish

In Spanish poetic tradition, several metrical structures have been used through time, and information about them can be found in works about Spanish metrics [Quilis, 1984, Tomás, 1995, Caparrós, 1999].

As noted in Quilis [1984], classification of verses in Spanish can be made according to: (1) the number of syllables, and (2) the stress of the last word. As these are the two factors that categorize verses in Spanish, Spanish poetry is accentual-syllabic.

Considering the number of syllables, *versos de arte menor* (minor art verses), *de arte mayor* (major art verses) and *compuestos* (composite verses) can be found. The main characteristic of minor verses is the agility they transmit. Light poetic compositions tend to use this meter. Minor meters comprise verses that contain from two up to eight syllables, as in this example from *La señorita del abanico* by Federico Garcia Lorca [Lorca, 2017] whose lines contain five syllables:

La señorita
del abanico,
va por el puente
del fresco río.
...

Major art verses are used to convey more thoughtful information, as they are longer verses. The verses must contain nine, ten or eleven syllables. In the poem *El estudiante de Salamanca* [Espronceda, 1999], José de Espronceda uses twelve syllables in his verses:[6]

...
Cual suele la luna tras lóbrega nube
con franjas de plata bordarla en redor,
y luego si el viento la agita, la sube
disuelta a los aires en blanco vapor:
...

The composite verses are made-up of two simple verses including a caesura or pause between them in a single line. This pause limits the use of syllable contractions, —i.e. synaloepha (explained below)—, as two syllables that are separated by a caesura cannot be contracted. These verses can contain twelve syllables (dodecasyllabic), fourteen syllables (alexandrine) or more. The poem about Gonzalo de Berceo in *Mis poetas*, by Antonio Machado [Machado and Cano, 1970], is composed of several verses, each of them containing two simple verses of seven syllables (and summing a total of fourteen syllables for each line):

...
Su verso es dulce y grave: monótonas hileras
de chopos invernales en donde nada brilla;
renglones como surcos en pardas sementeras,
y lejos, las montañas azules de Castilla.
...

The previous typology sets different kinds of verses based on their number of syllables. According to the last words stress, there can be verses whose last syllable is stressed (oxytone), verses whose penultimate syllable

[6]The second and fourth line contain eleven syllables. This phenomenon, with the same example, is explained below.

is stressed (paroxytone) and the last ones, whose antepentultimate syllable is prominent (proparoxytone). The lines that have their penultimate syllable stressed are the most common in Spanish poetry.

There are some metrical phenomena related to the last words stress. As explained and justified in Quilis [1967], the classification of verses according to the number of syllables can be changed slightly. For instance, a verse that has eight syllables and stress of the last word is on the penultimate syllable, will be considered an octosyllabic verse. Contrary to this, in oxytone verses, a syllable should be added, so a verse that contains seven syllables and has a stress on the last syllable, will also be considered octosyllabic. This happens in the poem *Rimas* by Gustavo Adolfo Bécquer [Bécquer, 2016]:

Saeta que voladora
cruza, arrojada al azar,
y que no se sabe dónde.
temblando se clavará;
...

The other special phenomenon is found in proparoxytone lines, in which the antepenultimate syllable is stressed. In this case, a verse that contains twelve syllables is considered a hendecasyllabic line. Proparoxytone verses are not frequent in Spanish poetry. Below an excerpt from the poem *Padre fray Adrián, ni el adriático* can be seen, by Cairasco de Figueroa [Durán, 1982]:

...
Estos con vos, por la región tritónica,
de las ieguas indómitas el piélago,
passaron hasta ver la plaia ibérica,
...

In this work, because of annotated corpus availability, I have focused only on a specific prosperous period, the Golden Age. In this period the main meter of poetry was the hendecasyllable, in which each verse had eleven syllables. The stress sequence of these lines is quite regular and usually the 10th syllable will be stressed (paroxytone verses). Other syllable positions are also stressed and this leads to a subcategorization of hendecasyllabic lines, which can be either

1. emphatic: 1st and 6th syllables are stressed

2. heroic: 2nd and 6th syllables are stressed
3. melodic: 3th and 6th syllables are stressed
4. or sapphic: 4th and 6th or 8th syllables are stressed

Apart from these predefined structures, others such as dactylic or iambic can be found, but they are not as common.

One of the challenges in Spanish poetry is the use of syllable contractions or synaloephas, which fit verses with more syllables into the predefined structures. As an example, let us recall a previous example:

...
Cual suele la luna tras lóbrega nube
con franjas de plata bordarla en redor,
y luego si el viento la agita, la sube
disuelta a los aires en blanco vapor:
...

In these lines, the first and third lines contain twelve syllables and the second and fourth ones eleven. The stresses found in the lines are repetitive:

...

Cual	*sue*	*le*	*la*	*lu*	*na*	*tras*	*ló*	*bre*	*ga*	*nu*	*be*
x	/	x	x	/	x	x	/	x	x	/	x

con	*fran*	*jas*	*de*	*pla*	*ta*	*bor*	*dar*	*la_en*	*re*	*dor*
x	/	x	x	/	x	x	/	x	x	/

y	*lue*	*go*	*si_el*	*vien*	*to*	*la_a*	*gi*	*ta*	*la*	*su*	*be*
x	/	x	x	/	x	x	/	x	x	/	x

di	*suel*	*ta_a*	*los*	*ai*	*res*	*en*	*blan*	*co*	*va*	*por:*
x	/	x	x	/	x	x	/	x	x	/

...

There is a recurrent pattern of beats, which is "`x/xx/xx/xx/x`" in the odd lines and "`x/xx/xx/xx/`" in the even lines. If the syllables are counted directly in the second line, there would be 12 syllables, but in order to fit in the poem's meter, the syllables from "bor.dar.la_en re.dor" must be joined.[7]

[7]Syllable union is expressed with the underscore symbol (_).

The same happens in the third and fourth lines, where some words have to be merged ("y lue.go si_el vien.to la_a.gi.ta" and "di.suel.ta_a los ai.res").

II.1.4 Tradition in Basque

In the Basque Country there exists a long-standing live performance tradition of improvising verses —a type of ex tempore composition and singing called *bertsolaritza* [Garzia et al., 2001]. In terms of written poetry, the first known printed book in Basque appeared in 1545. It was entitled *Linguae Vasconum Primitiae*, and it was written by Bernat Etxepare. This was a complete book containing several poems. A century later, Rafael Micoleta in 1653 [Micoleta et al., 1880], and Arnaut Oihenart in 1665 [Lafitte, 1967, Urkizu, 1994], attempted to formalize Basque poetic meter.

Current Basque poetry follows the syllabic metrical system and accents are not supposed to be taken into account. In spite of that, old Basque poets did not always use isosyllabic structures,[8] and the number of intelligible beats in those verses was the same, which suggests that the metrical unit in Basque was not the single syllable. Thus, older poetic tradition could be based on accentual or accentual-syllabic meter. Manuel Lekuona argued in an opening discourse [Lekuona, 1918, p. 29] that Basque verses are not built upon simple syllable count, but, rather, with a combination of counting syllables and counting plausible feet.

In this work I focus on the stress structure of the verses. More currently, some researchers have addressed this issue, some of them having contradictory opinions on the topic. Some experts claim that rhythm plays —or should play— an important role in Basque poetry, as it can be observed in the work Onaindia [1961]:[9]

...

Literatur guztiak dabez euren lege ta arauak, olerkigintzan bereziki; euskeran be naitaez izan bear. Lau gauza oneik beintzat gogotan artu bearrak doguz: 1) Igikera (ritmu); 2) etena (cesura); 3) neurria, ta 4) oskide edo azken amaitze bardiña (rima).

-

Any literary tradition has its own laws and rules, especially in poetry; Then, in Basque we must have them too. At least, we must take these four things into account: 1) Rhythm; 2) caesura; 3) meter, and 4) rhyme.

...

[8] Verses with the same number of syllables

[9] The original work is in Basque, and my own translation can be seen below.

Other researchers, however, state that in Basque stress does not play an important role in Basque. In fact, Nikolas Ormaetxea —Orixe— proposes an experiment in which he asks one hundred people to mark the stresses in a text. He ensures that these people will mark the stresses differently [Ormaechea, 1920]:[10]

...

Para probar lo poco sensible que es el acento vasco, inténtese colocar acentos gráficos en las silabas que uno crea acentuadas, encárguese el trabajo a cien personas de buen oido y en una página que se someta al análisis, se puede asegurar sin temor, que no habrá dos que coincidan.

-

In order to show the low sensitivity of Basque stress, try to mark stresses of a text. Ask one hundred people to mark the stresses and in a single page, I am completely sure that there will not be two equal analyses.

...

After reading what different researchers state about Basque stress, my insight is that if I ask a group of people that speak the same dialect in Basque, according to Zuazo-Zelaieta [1998], to tag a set of metrically regular poems, there should be a significant agreement.

Recent poetry in Basque is sung or written primarily on fixed metrical structures. These metrical structures are referred to as small and big meters, *neurri txikiak* and *neurri handiak*, respectively. "Neurri" is the Basque word for "meter", and "txiki" and "handi" refer to the size of the verse, being translated as small and big. This nomenclature is similar to the Spanish one.

On the one hand, small meters are composed of lines that must contain 7 and 6 syllables, in the odd and even lines, respectively. On the other hand, the lines in big meters have 10 and 8 syllables in the odd and even lines. Below, two examples can be seen of such meters, the first one by Xabier Amuriza[11] using a small meter and the second one by Unai Iturriaga[12] using a big meter:

Neu.rriz e.ta e.rri.maz	*Through meter and rhyme*
kan.ta.tze.a hi.tza	*to sing the word*
ho.rra_hor ze ki.rol mo.ta	*that is what kind of sport*
den ber.tso.la.ri.tza	*bertsolaritza is.*

[10] This work was published in Spanish and I also provide my translation below.

[11] `http://bdb.bertsozale.eus/en/orriak/get/4-zer-da-bertsoa`

[12] `http://bdb.bertsozale.eus/en/web/bertso/view/kf7lk`.

Ber.tso.la.ri.tzak ho.ri.xe dau.ka	That is what Bertsolaritza is
i.noiz gaiz.ki i.noiz on.gi	(we perform) sometimes bad, sometimes well
e.ta tar.te.an gal.de.ra_i.kur bat	and sometimes a question mark,
zer den ez da.ki.gun ho.ri.	what we do not know.
No.rai.no nai.zen a.zal.du nahi.an	I wanted to show how far I could reach,
e.to.rri naiz e.ta to.ri!	I came for that but I could not show it!
E.san ez dut ba ber.tso.a_hau.xe da:	That's a Basque poem,
ten.te_i.raun e.do e.ro.ri.	Stand up or fall down.
Ha.le.re mai.te nau.zu.e.la.ko	In spite of that, as you (listeners) love me
es.ke.rrik as.ko da.no.ri.	thank you all.

The number of lines expresses the name that the whole meter will receive. For example, a poem with small meter and eight lines will be called small of eight, or *Zortziko txikia*.[13] The poems containing eight or ten lines are the most common ones, although others, more special ones, are becoming popular. These structures are very diverse; some of them are based on classical songs by Basque musicians, such as Xabier Lete[14] or Mikel Laboa,[15] and others are based on international songs, such as "*Redemption*",[16] by Bob Marley or "*Blowing in the wind*",[17] by Bob Dylan.

II.2 Automatic scansion of poetry

Automatic scansion of poetry has attracted attention from numerous scholars for years, and in the following section some of them are cited. Some works perform a statistical analysis, like Hayward [1991] and Hayes et al. [2012]. Others use linguistic knowledge acquired by making observations in different kinds of poetry and they use this knowledge to generate some rules for the assignment of stress. Over the last few years, with the development of the Machine Learning based methods, supervised machine learning systems are appearing as will be observed below.

[13] *Zortzi* is the word in Basque for the number "eight" and *txiki* is "small".

[14] `http://bdb.bertsozale.eus/en/web/doinutegia/view/2149-izarren-hautsa-egun-batean`

[15] `http://bdb.bertsozale.eus/en/web/doinutegia/view/2932-eguzkiak-urtzen-du-han-goian-a`

[16] `http://bdb.bertsozale.eus/en/web/doinutegia/view/3157-azken-tragotxo-hori-hartu`

[17] `http://bdb.bertsozale.eus/en/web/doinutegia/view/3163-geldirik-ezin-naizela-egon`

II.2.1 Rule-based scansion

Logan [1988] documents a set of programs to analyze sound and meter in poetry. This work falls in a general genre of techniques that attempt to analyze the phonological structure of poems following the generative phonological theory outlined by Chomsky and Halle [1968] and described by Brogan [1981].

Gervas [2000] proposed a logic programming approach for the analysis of Spanish poems and evaluated such approach in a set of Spanish Golden Age sonnets. This system used a logic programming approach (Definite Clause Grammars) for the syllabification and various rules to assign stress to syllables. Apart from that, it performed rhyme analysis on the input poems.

Scandroid is a program that scans English verse written in either iambic or anapestic meter, designed by Charles O. Hartman [1996, 2005]. The source code is publicly available.[18] The program can analyze poems and check if the predominant stress pattern is iambic or anapestic. However, if the input poem's meter is not one of those two, the system forces each line into one of them.

AnalysePoems is another tool for identification of metrical patterns, written by Plamondon (2006). In contrast to other programs, its main goal is not to perform perfect scansion (p. 128-129), but to identify the predominant meter of a poem. The program also checks the rhyme scheme found in the input poem. It is reportedly developed in *Visual Basic* and the .NET framework; however, neither the program nor the code appear to be available.

Calliope is a similar tool, built on top of Scandroid by Garrett McAleese [2007]. It is an attempt to take advantage of syntactic information in order to improve scansion. The program does not seem to be freely available.

Bobenhausen and Hammerich present a tool[19] that performs general poetry analysis including scansion in German [Bobenhausen, 2011, Bobenhausen and Hammerich, 2015]. The system, called *Metricalizer*, analyzes several aspects of an input poem, such as the prosodic structure, rhymes and classification of a poem according to the previous features. Its output can be saved in TEI format.[20] Metricalizer has been used in other projects, such as Baumann and Meyer-Sickendiek [2016], Kraxenberger and Menninghaus [2016].

In Navarro-Colorado [2015] a hybrid approach to Spanish meter is presented. The author proposes a chain that performs syllabification, POS-

[18] `http://oak.conncoll.edu/cohar/Programs.htm`

[19] The tool is available online at `http://www.metricalizer.de/`

[20] A commonly used format for encoding textual data.

tagging and blending or segmenting syllables according to synaloephas, dieresis or syneresis. The system is hybrid as it performs scansion in a rule-based fashion and statistical information is used for the disambiguation of possibly ambiguous cases. It also performs clustering of similar poems by using topic modeling and K-means clustering.

Of the current efforts, the work Delmonte [2016] should be underlined, as Shakespeare's sonnets are analyzed from different points of view, for example, the concreteness/abstractness of words, scansion, rhymes, alliterations and so on. Scansion is performed using a set of rules and a syllable list to divide the words into syllables and assign the stress.

II.2.2 Data-driven scansion

In Hayward [1991] one of the goals of analysis was to see whether it would be possible to differentiate among the metrical patterns developed by individual writers. They also had a goal of analyzing the stylistic differences among periods, which is why they collected a corpus with several poets from several periods. In order to analyze poetry in the above senses, they built a model based on what they called Parallel Distributed Processing (PDP) [Rumelhart et al., 1988b], which nowadays is better known as Neural Networks. They created a Neural Network that had 5 inputs that encode different information about the poetry line to be analyzed. These 5 features include intonation, lexical stress, prosodic information, grammatical structure and the interpretation of the significance of a word within a poem (subjective). Each of the 5 inputs had ten values, which encode feature values for each of the syllables of the line. Hayward analyzed ten different poets representing different periods, two of whom were from the renaissance, another two represented neo-classical verse, three from the romantic period, two Victorian and a last twentieth-century American writer. With this information, Hayward analyzed their poetic work and he argues that the computerized connectionist model of poetic meter was successful in determining significant differences among the analyzed poets.

Greene et al. [2010] uses statistical methods in the analysis of poetry. For the learning process, *The Sonnets* by Shakespeare was used, as well as a number of other works freely available online.[21] Weighted finite-state transducers were used for stress assignment. As with the other documented projects, there is no implementation to review.

[21] `http://www.sonnets.org`

Hayes et al. [2012] propose in their article a new approach to analysis in metrics. Their research is built upon two main works: Generative metrics [Halle and Keyser, 1971] and Maxent Grammars [Hayes and Wilson, 2008] for the analysis of phonotactics. In this work they propose a set of constraints that can appear in a verse line, and according to the number of times each of these constraints is not fulfilled and according to the weight of each constraint, they calculate the metricality of a line. The proposed method is like a Logistic Regressor, in which the features are the number of times each constraint is violated. Then they learn the weights for each of the constraints by Maximum Likelihood to maximize the predicted probability of all the lines in the corpus. The selected features are constraints that they present in section 5. Some of the constraints include elements related to prosodic hierarchy, and their boundaries. Others refer to the relative stress between syllables, as in iambic pentameter, it is expected that rising and falling syllables be in specific positions. In the model presented in Hayes and Wilson [2008] they proposed a method for learning the constraints by maximizing the probability of the corpus. One of the interesting results of this work was the tagged corpus that they created, which was composed using Shakespeare's *Sonnets* and Milton's *Paradise Lost* (eighth and ninth books), manually tagged by using a four-level notation traditionally used in generative metrics [Halle and Keyser, 1971].

Estes and Hench [2016] is a current work that makes use of supervised learning tools in order to metrically analyze poems written in Middle High German. Middle High German poetry is a hybrid between qualitative and quantitative verse, which means that both the length and the stress of syllables are taken into account for patterning in the lines. In order to perform supervised learning, they use a corpus of 825 manually annotated lines, which are marked by the authors. The features that are used for the learning include the position within the line, length of the syllable in characters, some specific characters of the syllable, next syllables first two characters, previous words last two characters, syllable weight and length and word boundaries. The model used is a Conditional Random Field and they report to achieve an F-Score of 0.894 on 10-fold cross-validated development data and 0.904 on held-out testing data.

II.2.3 Automatic poetry analysis

In the following lines I present some systems that perform general poetry analysis, modeling elements like semantics, syntax, sound devices, to name a few.

In Kaplan and Blei [2007], the authors made a computational stylistic analysis of American poetry. By projecting poems into a multi-layered latent structure and computing distances among these representations, the authors analyze and classify American poetry (by style and author). They analyze orthographic (word count, number of lines, number of stanzas, average line length (in words), average word length, average number of lines per stanza and the frequency), syntactic (part of speech frequencies) and phonemic features (rhymes, alliteration, assonance and consonance). These authors visualize the embedded poems by using Principal Component Analysis (PCA). They demonstrate that their method delineates poetry style better than the traditional word-occurrence features that were used in typical text analysis algorithms before.

In 2012, Kao and Jurafsky make use of computational methods to compare the stylistic and content features among award-winning poets and amateur poets. The features that they use in order to analyze poetry are diction, sound devices, affect and imagery. Unfortunately, it appears that they do not use meter as an informative feature, although it is a sound device. They make quite an extensive analysis of the stylistic differences between award-winning poets and amateur ones.

In McCurdy et al. [2015], the authors present a formalism to describe rhyme in poetry, which is represented as rhyming templates, in a similar way that was done in our previous work Agirrezabal et al. [2012b]. The difference from our work was that in the work by McCurdy et al. stress values are taken into account and a text written in Arpabet[22] format must be given as input. Moreover, apart from the formalism, they also present a tool to analyze sonic devices in poetry, called *RhymeDesign*.

II.3 Sequence modeling

In the present study I analyze poems and obtain information about their rhythm. As the goal is to assign stresses to syllables, the problem can be treated as a sequence modeling task, where the elements within a sequence will be given a tag. In the *Natural Language Processing* (NLP) literature, several systems have been proposed for this task. This section gives an overview of some typical approaches to sequence modeling in NLP along with some examples and applications.

A supervised learning problem is a Machine Learning task where some data and its target values are given [Friedman et al., 2001]. The input data is

[22]The formalism used in the CMU pronunciation dictionary [Weide, 1998].

a set of examples or instances which have an output. These output values can be a continuous value, regression, or a set of specified k values, classification. Each instance of the input data is represented by a vector of numeric values and these methods attempt to learn a function that will perform the mapping between input examples and target values.

In some cases, nevertheless, the predictions cannot be made simply by looking at local values or local elements, as the global structure of the output is important [Daume, 2006, Sutton and McCallum, 2011, Graves, 2012]. In such cases, the problem can be tackled as a structured prediction problem, in which a sequence of input feature vectors is given and a (hopefully) optimum result is predicted. Structured prediction is a set of machine learning techniques, whose aim is to predict structured objects, rather than single outputs, like in previously mentioned classification or regression problems.

It is noteworthy to understand the importance of doing predictions jointly, and not just independently. The main advantage is that they calculate the optimal resulting sequence for each input. Let's suppose we want to create a very simple part-of-speech tagger and we want to assign POS-tags to each of the word according to their joint probability. In order to do it a simple table can be used, where for each word in a vocabulary, includes the probability of having such POS-tag given the word. This information can easily be calculated from tagged corpora.

Word	...	DET	MOD	NN	PRP	VB	...
a	...	0.95	0.0	0.05	0.0	0.0	...
buy	...	0.00	0.0	0.015	0.0	0.985	...
can	...	0.0	0.7	0.3	0.0	0.0	...
car	...	0.0	0.0	1.0	0.0	0.0	...
have	...	0.0	0.0	0.03	0.0	0.97	...
I	...	0.0	0.0	0.0	1.0	0.0	...

Imagine that we are given the following sentence,

I	have	a	car
PRP=1.0	VB=0.97	DET=0.95	NN=1.0
	NN=0.03	NN=0.05	

whose POS-tags can be easily assigned by considering the probability of each tag, given the word.

I+PRP have+VB a+DET car+NN

In this example, the assignment of the part of speech is fairly straightforward, as we only have to check the probabilities between the input and output variables. One of the most challenging problems in Natural Language Processing is ambiguity, and ambiguity can make a POS-tagger fail when predicting values. For instance, if the input sentence is

I	can	buy	a	can
PRP=1.0	MOD=0.7	VB=0.985	DET=0.95	MOD=0.7
	NN=0.3	NN=0.015	NN=0.05	NN=0.3

by assigning the tags in the same way as before, we will get as a result

I+PRP can+MOD buy+VB a+DET can+MOD

which, obviously, is not correct as the second occurrence of *can* is not a modal, but a singular noun. In structured prediction models, the transitions between output elements are also modeled. Obviously, the probability of having a noun after a determiner is considerably higher than having a modal. In this way, structured prediction systems learn more complex but more reliable models.

II.3.1 Greedy sequence tagging/modeling - Sequence tagging as classification

Sequence tagging is done greedily when predictions are made locally, i.e. each word —or each element in the sequence— is treated independently. It is typical the use of previously made predictions so as to model the complex dependencies among the outputs [Sutton and McCallum, 2011]. For example, in figure II.1, each prediction is only dependent on the current word and the previous prediction (forward tagging). In order to calculate the tag `VB` that should be assigned to the word *chases*, its probability has been estimated according to the current word (*chases*) and the previous prediction (`NN`). In figure II.2 the model performs a similar function by using the next prediction (Backward tagging).

In the previous two cases, for each instance being predicted, there are two different elements, i.e. attributes, which should give enough information for the current prediction. These are the current word and the previous or next prediction.

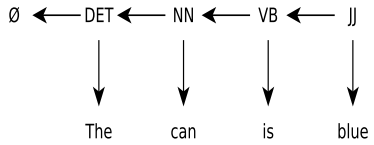

Figure II.1: Forward tagging using current element and previous prediction.

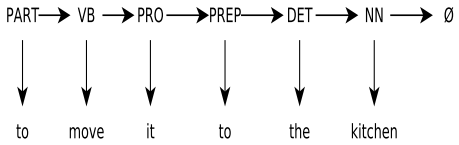

Figure II.2: Backward tagging using current element and next elements prediction.

II.3.2 Structured prediction

In figure II.1 and II.2, it can be seen that if one of those predictions is not done correctly, its error will be propagated throughout the whole sequence, as that prediction is used to estimate the next elements POS-tag. This is called *error-propagation*. Structured prediction systems tackle this problem by generating a set of probabilities for the sequence. Once a set of probabilities is calculated for each element the optimal one over all possible output sequences is found. Consider the following sentence

I can buy a can

whose possible POS-tags[23] would be

I	can	buy	a	can
PRO=1.0	MOD=0.7 NN=0.3	VB=1.0	DET=1.0	MOD=0.7 NN=0.3

In order to calculate the optimal resulting sequence, we need to know the probabilities of the output sequences. Below the informative probabilities are included and, for the sake of simplicity, the other transition probabilities will be assumed to be 1.0.

PRO - MOD	0.98
PRO - NN	0.02
DET - MOD	0.08
DET - NN	0.92

[23]This is obviously an artificial example and its probabilities have been made up.

At this point, all the possible outputs have to be generated and the most probable one has to be chosen. So that to calculate the whole sentences probability, this probability must be calculated for each element in the sequence:

$$\underbrace{P(postag[t]|word[t])}_{Emission\_probability} x \underbrace{P(postag[t]|postag[t-1])}_{transition\_probability}$$

1. PRO MOD VB DET MOD $\rightarrow (1.0 \times 1.0) \times (0.7 \times 0.98) \times \ldots \times (0.7 \times 0.08) = 0.038416$
2. PRO MOD VB DET NN $\rightarrow 1.0 \times 0.686 \times 1.0 \times 1.0 \times 0.276 =$ **0.189336**
3. PRO NN VB DET MOD $\rightarrow 1.0 \times 0.006 \times 1.0 \times 1.0 \times 0.056 = 0.000336$
4. PRO NN VB DET NN $\rightarrow 1.0 \times 0.006 \times 1.0 \times 1.0 \times 0.276 = 0.001656$

This was a simplistic view of a more complex problem. Usually the number of possible sequences and the possible POS-tags is larger, so calculating such probabilities is computationally complex. In order to do it efficiently an algorithm called *Viterbi* is used.

II.3.3 Scansion as a sequence modeling problem

In this section until now, I have reviewed the two main approaches when sequence-to-sequence problems must be solved. The two approaches were presented using a typical task in Natural Language Processing, which is POS-tagging.

The problem I am facing in this work is similar, as, given an input sequence, an output sequence that best fits has to be calculated. Given a sentence (syllabified or not) a sequence of stresses must be returned. Let's suppose that the input sentence (verse) is this, from "*To Autumn*" by John Keats:

To swell the gourd and plump the hazel shells

On the one hand, if the sentence is previously syllabified, then the model may have to learn a syllable to stress mapping, for instance:

Input:	to	swell	the	gourd	and	plump	the	ha	zel	shells
Output:	x	/	x	/	x	/	x	/	x	/

On the other hand, if the words have not previously syllabified:

Input:	to	swell	the	gourd	and	plump	the	hazel	shells
Output:	x	/	x	/	x	/	x	/x	/

In the first case, the possible output set is `/` and `x`. However, in the second case, the set should also include the pattern `/x`, apart from `/` and `x`. Then, the output set, or dictionary, is bigger.

CHAPTER III

Annotated corpora of verse

In order to scan poetry automatically, it is important to have access to metrically annotated poetry. Unfortunately, there are not very many annotated poetry corpora available. To the extent of my knowledge, these are the most significant repositories of annotated poetry corpora:

- For Better For Verse (4B4V)[1] [Tucker, 2011]
- ReMetCa[2] [González-Blanco García and Rodríguez, 2013]
- Corpus de Sonetos del Siglo de Oro[3] [Navarro-Colorado et al., 2016]
- The Corpus of Czech verse[4] [Plecháč and Kolár, 2015]

Then, I tried to collect written corpora in different languages that included metrical information about poems. As it may be seen in later sections, three collections of poetry in English, Spanish and Basque are used. The English work is used as a basis for all the experiments, and then the parameters learned from it are used in Spanish and Basque. The main reasons for using the English and Spanish corpora is not only the availability of the corpus but also my linguistic competence in both languages. I am working with the Basque language in order to make a contribution in this understudied subject. The collection in Basque was not annotated metrically, so this is the first proposal of an annotated corpus.

[1] `http://prosody.lib.virginia.edu/`
[2] `http://www.remetca.uned.es/`
[3] `https://github.com/bncolorado/CorpusSonetosSigloDeOro`
[4] `http://www.versologie.cz/en/kcv.html`

III.1 Encoding of poetry

In this work the encoding format that was first used in the 4B4V corpus is followed, the Text Encoding Initiative[5] (TEI) guidelines [TEI Consortium, 2008]. Text Encoding Initiative is a consortium responsible for developing and maintaining the standards for encoding textual data. The guidelines they propose are widely followed by libraries, museums, publishers, and individual scholars. Apart from being widely used, it has specific features for encoding information about poetry[6] —documented in the Verse module—, such as rhyme, meter, caesuras and so on.

One of the advantages of annotating poetry in a unified manner is that the information can be extracted easily. Consider this excerpt from Lewis Carroll's "*Phantasmagoria and other poems*" [Carroll, 1869]:

Away, fond thoughts, and vex my soul no more!
Work claims my wakeful nights, my busy days,
Albeit bright memories of the sunlit shore
Yet haunt my dreaming gaze.

This is the forth stanza from a poem that appears at the beginning of the book. Following the TEI guidelines, the lines have to be placed between `l` tag and whole stanzas between the `lg` tags. Each of these elements can incorporate some attributes, e.g., `@n`, which will mark the line or stanza number. This information is shown in listing III.1.

Listing III.1: Marking line groups and lines.

```
<lg n="4">
<l n="1">Away, fond thoughts, and vex my soul no more!</l>
<l n="2">Work claims my wakeful nights, my busy days,</l>
<l n="3">Albeit bright memories of the sunlit shore</l>
<l n="4">Yet haunt my dreaming gaze.</l>
</lg>
```

In listing III.2, it can be seen what the stanza looks like when stress and rhyme information is added. Readers should be aware of the interest of marking the rhyme, as the rhyme of some words cannot be inferred directly without having knowledge about the language in question, e.g., *days* and *gaze*. In the case of stress, information can be incorporated in the `l` element by using the attributes `@met` and `@real`. The first attribute, `@met`, includes the metrical structure that the line follows and the second one, `@real`, shows the actual realization when the poem is read aloud. In this work, this second

[5] `http://www.tei-c.org`
[6] `http://www.tei-c.org/release/doc/tei-p5-doc/en/html/VE.html`

parameter (`@real`) is used for learning and evaluation. The stress values in these attributes are marked with the `+` and `-` symbols, as previously mentioned in subsection II.1.1.

Listing III.2: Marking stress and rhyme information.

```
<lg n="4">
<l n="1" met="-+-+-+-+-+" real="-+-+-+-+-+">
Away, fond thoughts, and vex my soul no <rhyme label="a">
    more</rhyme>!
</l>
<l n="2" met="-+-+-+-+-+" real="++-+-+-+-+">
Work claims my wakeful nights, my busy <rhyme label="b">
    days</rhyme>,
</l>
<l n="3" met="-+-+-+-+-+" real="-+++---+-+">
Albeit bright memories of the sunlit <rhyme label="a">shore
    </rhyme>
</l>
<l n="4" met="-+-+-+" real="-+-+-+">
Yet haunt my dreaming <rhyme label="b">gaze</rhyme>.
</l>
</lg>
```

But we can go forward. Syllables can be marked too. The three corpora incorporate information about syllable division, but the public version of the Spanish corpus does not show it. In the 4B4V corpus, as the main dividing unit is the foot, lines are divided in `seg` elements, which represent each foot and syllables are divided using the syllable boundary (`sb`) element. In listing III.3 you can see the third line of the previous example divided into syllables.

Listing III.3: Marking feet and syllable boundaries.

```
...</l>
<l n="3" met="-+-+-+-+-+" real="-+++---+-+">
<seg>Al<sb/>beit</seg> <seg>bright me<sb/></seg><seg>mories
    of</seg> <seg>the sun<sb/></seg><seg>lit <rhyme label="
    a">shore</rhyme></seg>
</l>...
</lg>
```

When foot division is not trivial, as in Basque, syllables are marked with a slight difference. In listing III.4, these two lines are encoded into XML:

...
huntzak egin oihu:
akherra hor heldu.
...

Listing III.4: Marking feet and syllable boundaries in the Basque corpus.

```
<l n="3" met="" real="+--+-+">
  <seg type="syll" targetId="w13">hun</seg>
  <seg type="syll" targetId="w13">tzak</seg>
  <seg type="space"> </seg>
  <seg type="syll" targetId="w14">e</seg>
  <seg type="syll" targetId="w14">gin</seg>
  <seg type="space"> </seg>
  <seg type="syll" targetId="w15">oi</seg>
  <seg type="syll" targetId="w15">hu</seg>
  <seg type="punct" targetId="w16">:</seg>
</l>
<l n="4" met="" real="+--+-+">
  <seg type="syll" targetId="w17">ak</seg>
  <seg type="syll" targetId="w17">he</seg>
  <seg type="syll" targetId="w17">rra</seg>
  <seg type="space"> </seg>
  <seg type="syll" targetId="w18">hor</seg>
  <seg type="space"> </seg>
  <seg type="syll" targetId="w19">hel</seg>
  <seg type="syll" targetId="w19">du</seg>
  <seg type="punct" targetId="w20">.</seg>
</l>
</lg>
```

This information, among other data, is available in all the corpora that is used in this work. Having information encoded in such a way, allows computers to access digitized data and get information that could not be extracted (or would be hard to extract) from raw text.

Encoding poetry in different languages

A problem that was encountered when creating the Basque annotated corpus, was that the 4B4V corpus' main dividing unit is the foot. The poems are tagged according to the predominant foot used in the poem. But, as the foot division in Basque is not so well-known (trivial), I decided to make the syllable as the main smallest constituent unit of ordinary matter, as it could be seen in listing III.4. The same is done in the Spanish corpus, as can be seen in Navarro-Colorado et al. [2015].

Another relevant issue relates to the goal of the corpora that is used to create the models. For example, the English corpora was created to be used in an online tutorial to teach people to scan poetry in English. The focus of the corpus in Spanish was to create ML models to analyze poetry, and as a

result, ambiguity cases have been resolved in a consistent way so that ML models work better.

III.2 English poetry corpus

As the gold standard material for training the English metrical tagger, the previously mentioned corpus, For Better For Verse (4B4V), from the University of Virginia[7] [Tucker, 2011] was used. 4B4V is an interactive on-line website to train people in the scansion of English poetry in traditional meter. The site has been brought by the Scholar's Lab at the University of Virginia and it acts as an interface between the user and the tagged corpus, which can be easily interpreted by the machine. The poems can be downloaded from a public repository on GitHub.[8,9] The website shows a poem from the corpus and the user can mark each syllable's stresses in the upper part of each line, unstressed (u) or stressed (/). The feet can be marked throughout the line, marking it between syllables. At the right side of each line, there are three buttons, two of them for checking the marked stresses or feet and the last one for setting the meter of the line.

The basic metrical unit is the foot and it is represented using the `seg` element within each line. The rhythmic structure is shown by a sequence of stressed/unstressed syllables for each line. The author of the 4B4V corpus says to have a rising rhythm preference [Tucker, 2011, sec. 3].

Statistics

The entire collection is composed of 78 poems and approximately 1100 lines. In table III.1[10] the obtained number of poems per period can be seen. Among the 34 different authors, 31 are men and 3 are women. There is also a bias in favor of British poets, as 31 out of 34 are British, two American and one Irish. The collection of poems is rather homogeneous, the predominant meter of the poems being iambic (91.38% of the lines). The remaining 8.62% lines use trochaic (3.12%), anapestic (4.12%), dactylic (1.19%) or spondaic

[7] `http://prosody.lib.virginia.edu/`

[8] `https://github.com/waynegraham/for_better_for_verse/tree/master/poems`

[9] When the corpus was first downloaded, in 2013, 54 poems were downloaded, getting them manually from the interactive website, by crawling the web. At a version downloaded in November 30, 2015 from GitHub, 78 poems were available. This explains the difference of results from Agirrezabal et al. [2013c, 2016b] to the ones presented here.

[10] Although the exact writing year is unknown, in this work, the poem *Westron wynde* is considered a poem written between 1500 and 1599.

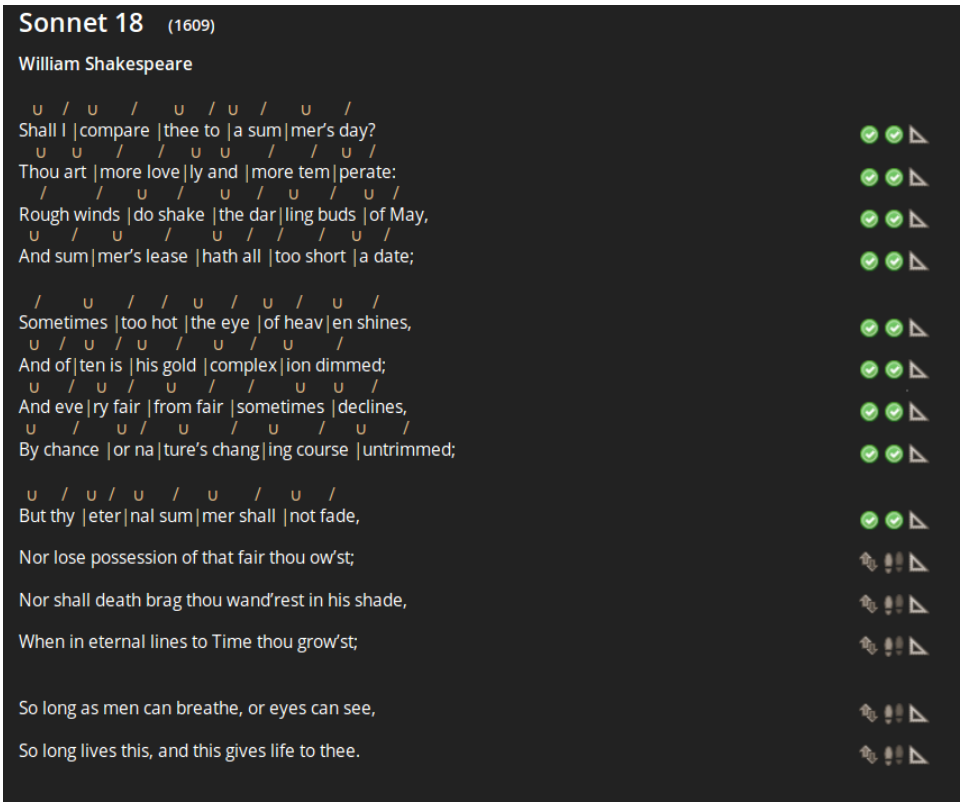

Figure III.1: Guessing the stresses in Shakespeare's sonnet no. 18

(0.18%) meters. Sometimes several analyses are given as correct, as there is ambiguity when performing scansion. About 10% of the lines are ambiguous and almost all of them have two alternative analyses, although they can have three, four or five analyses.

Adaptation

Some of the annotated files were duplicated, and in some cases there were two files with the same poem but not with the same additional information. In such cases, all duplicated files were checked manually and the one that seemed to be the most accurate was taken. Additionally, in seven poems small changes had to be made, such as making the spaces be at the end of each segment (Unless that was done, the corpus parser would raise an error).

Listing III.3 shows an example of how a verse from the English poetry corpus

	#Poems
1500-1599	6
1600-1699	18
1700-1799	6
1800-1899	32
1900-1999	16
TOTAL	78

Table III.1: Amount of poems per period of 100 years in the English corpus.

is annotated in 4B4V.

III.3 Spanish poetry corpus

For the Spanish language a corpus of Spanish Golden-Age Sonnets[11] [Navarro-Colorado et al., 2015, 2016] is used, which can also be downloaded from GitHub.[12] This is a collection of poems from the 16th and 17th century, better known as The Spanish Golden Age.[13] According to the authors, when collecting this corpus, the goal was not to gather canonical writers, but to collect the widest range of writers of the period, so as to create a representative corpus. In this collection, there is some metadata including information about the sonnet (author, title and encoding) and also information about the metrical annotation status, whether the stresses have been manually checked or not.

A difference regarding the poetry corpus in English is about the basic metrical unit. In the Spanish corpus each verse is marked with a sequence of stresses, which are encoded using the two-level marking in the same way as previously.[14] Then, in this case, feet are not marked as they are in the 4B4V corpus.

[11] Downloaded on September 1, 2016.

[12] `https://github.com/bncolorado/CorpusSonetosSigloDeOro`

[13] Although using a corpus that covered from the 16th until the 20th century would be better, such corpus was not available. Performing further experiments with the medieval Spanish corpus ReMetCa [González-Blanco García and Rodríguez, 2013] is interesting.

[14] Stressed and unstressed syllables are represented with the `+` and `-` symbols, respectively.

Statistics

The corpus is composed of 5,078 sonnets, which add up to 70,000 lines, although not all the corpus is manually checked. From all authors, there is only one woman, Juana Inés de la Cruz, and almost all of the authors were Spanish. Of all the authors, one was from Brazil, another one from Sardinia and a last one from Mexico (Juana Inés de la Cruz).[15]

At present, the portion that has been manually checked is composed of approximately 135 sonnets and almost 2,000 lines. These poems were written by seven different well-known authors,[16] all of them men and Spanish. This portion is used to train and test the supervised models.

Adaptation

The most common meter used in Spanish poetry is the hendecasyllable, where each verse should contain eleven syllables. In spite of that, not all verses contain eleven syllables in a strict sense. Sometimes syllable contractions are performed and two syllables are realized within the time of just one syllable. This device is known as synaloepha. In the 4B4V corpus, each syllable is marked with a level of stress, but in the Spanish corpus a stress sequence is set for each verse. In some cases, the number of syllables and stresses is not coherent. In order to deal with this, a heuristic was created to balance them, whose algorithm can be found in appendix A. Broadly speaking, when synaloephas appear, the heuristic adds unstressed syllables trying to keep especially the lexical stresses. Because of this decision, the results might be affected. Scansions will have an important bias towards lexical stresses and as the place of the addition of unstressed syllables is not regular, the resulting sequences will not be as regular as before. It is expected that this will especially affect the structured prediction systems, as they model the global structure of the output.

[15] There are some of them that lived out of Spain, but were considered Spanish.

[16] Miguel de Cervantes, Fernando de Herrera, Garcilaso de la Vega, Luis de Góngora, Gutierre de Cetina, Félix Lope de Vega and Francisco de Quevedo.

Listing III.5: Spanish poetry corpus example. The corpus originally included only the met attribute and using the mentioned heuristic the real attribute was added so that each syllable is mapped to a stress value.

```
1          <ns0:lg type="terceto">
2             <ns0:l met="---+++-+-+-" n="9" real="
                  ---+-++--+--+-">Con El Isidro un cura de una
                   aldea,</ns0:l>
3             <ns0:l met="---+---+-+-" n="10" real="
                  ---+---+-+-">con Los Pastores de Bel&#233;n
                  Burguillo,</ns0:l>
4             <ns0:l met="-----+-+-+-" n="11" real="
                  -----+-+-+-">y con La Filomena un idiota.</
                  ns0:l>
5          </ns0:lg>
```

An example of the Spanish poetry corpus can be seen in listing III.5. The original corpus only contained the `@met` attribute and for this work, the `@real` attribute was added, which was calculated using the heuristic mentioned above and explained in appendix A.

III.4 Basque poetry corpus

At the very beginning of this work, there was no metrically analyzed corpus for Basque. Therefore, I am attempting to propose the first metrically analyzed corpus for the Basque language. In order to create it we had to make a poem selection that would be included. Three steps were followed in order to create a metrically annotated corpus:

1. Selection of anthology
2. Selection of authors
3. Tagging of corpus

III.4.1 Selection of anthology

The first step was to take a previously made poetic anthology so as to facilitate the process of selecting which poems to include. Among different options, below some of the checked collections can be seen:

- *Mila euskal-olerki eder (Aita Onaindia)* [Onaindia, 1976]

- *Fantasía y realidad: Selección literaria vasca* [Lafitte and Barbier, 1967]
- Gure poesia: Juan Kruz Igerabideren antologia [Igerabide, 1997]
- *Antología poética vasca* [Arzallus, 1987]
- *Euskal poesia kultoaren bilduma (1880-1982)* [Amenabar, 1983]
- *Antología de la Poesía Vasca / Euskal Poesiaren Antologia [Aldekoa, 1993]*
- *Poesía Vasca. Antología bilingüe (Patrizio Urkizu)* [Urquizu Sarasua, 2009]

Of these anthologies, one was selected. The goal was to find a corpus that covered (if possible) the same period as the English poetry corpus. The intention was also to have poems with diverse topics, and because of that, some collections that only included religious topics in the poems were discarded. The corpus should also be digitized so that so that to avoid the tedious work of transcription of poems or with scanning and Optical Character Recognition (OCR). After analyzing the pros and cons of each option, the anthology that Patrizio Urkizu made in his book "*Poesía Vasca. Antología bilingüe*" was considered the most appropriate one. Another useful characteristic of this collection was that for each poem it included a translation in Spanish.

As with most poetry corpora, the majority of authors in this corpus are male, with 73 male and 4 female poets, summing a total of 77 poets. There are other 3 anonymous poems. In the poetry collection there were poets from different regions in the Basque country. In table III.2 the distribution of the authors' regions can be seen. The authors are distributed among the Basque regions Gipuzkoa, Araba, Biscay, Navarre and the northern region of the Basque Country.

III.4.2 Selection of authors

The whole selection of poems was too large for the first attempt of creating an annotated corpus. The collection's distribution was not evenly distributed over periods of time (centuries). In the 16th, 17th and 18th century there were 30 poems (10 poems for each century) but in the last two centuries, there were far more poems. The solution was to analyze all the poems in the first three centuries. In the last two centuries, as a rule of thumb, it was decided to choose only poets who had more than one work in the corpus,

Region	No. of poets
Gipuzkoa	35
Northern region	25
Bizkaia	11
Nafarroa	4
Araba	2

Table III.2: Distribution of the authors among regions in the Basque Country (whole corpus).

Region	No. of poets
Gipuzkoa	15
Northern region	21
Bizkaia	2
Nafarroa	2
Araba	1

Table III.3: Distribution of the authors among regions in the Basque Country (selected corpus).

assuming that authors with more than one contribution were perhaps better. For each of the selected poets, only one poem was chosen, the most famous one, and if this was not evident the choice was made randomly.

There are 41 poets in this selection and only one of them is a woman. This selected portion includes approximately 54 poems and 2400 lines.

III.4.3 Annotating the corpus

After choosing the corpus and the poems, the next task was to convert the raw text to a machine readable (and interchangeable) format. In order to identify the words in the corpus the tokenizer that is used in the IXA pipeline [Agerri et al., 2014] was used, which is available on GitHub.[17] Let's take as an example the poem excerpt that we used at the beginning of this chapter, which is the poem *Akhelarre* by Jules Moulier -*Oxobi*-:

...
huntzak egin oihu:
akherra hor heldu.
...

[17] `https://github.com/ixa-ehu/ixa-pipe-tok`

In listing III.6, you can see the analysis produced by the Ixa pipeline for the whole poem, including only the previous excerpt.

Listing III.6: Automatically processed poem.

```
<?xml version='1.0' encoding='UTF-8'?>
<NAF xml:lang="eu" version="2.0">
<nafHeader>
 <linguisticProcessors layer="text">
 ... (TOKENIZER/POS-TAGGER/NER-TAGGER/...)
 </linguisticProcessors>
</nafHeader>
 <text>
  ...
  <wf id="w13" sent="1">huntzak</wf>
  <wf id="w14" sent="1">egin</wf>
  <wf id="w15" sent="1">oihu</wf>
  <wf id="w16" sent="1">:</wf>
  <wf id="w17" sent="1">akherra</wf>
  <wf id="w18" sent="1">hor</wf>
  <wf id="w19" sent="1">heldu</wf>
  <wf id="w20" sent="1">.</wf>
 </text>
 <terms>
  <term id="t13" lemma="huntz" morphofeat="NC0NP000" pos="N
      " case="IZE ARR BIZ- ABS NUMP MUGM @OBJ @PRED @SUBJ">
   <span>
    <target id="w13"/>
   </span>
  </term>
  <term id="t14" lemma="egin" morphofeat="VM000T00" pos="V"
       case="ADT PNT ANB MDNC NOR_NORK NR_HURA NK_HIK-NO @+
      JADNAG">
   <span>
    <target id="w14"/>
   </span>
  </term>
  <term id="t15" lemma="oihu" morphofeat="NC000000" pos="N"
       case="IZE ARR BIZ- ZERO @KM&gt;">
   <span>
    <target id="w15"/>
   </span>
  </term>
 </terms>
```

After tokenization, a Python module combines the tokenized text in NAF format[18] [Fokkens et al., 2014] —that is, the representation from list-

[18] http://wordpress.let.vupr.nl/naf/

ing III.6— and the poem itself (in raw text format) to create a document, where metrical information will be added manually. Additionally, a finite-state technology based syllabification system [Hulden, 2009] is incorporated, and then, in the resulting document syllables are divided. The syllabification procedure is based on Hulden [2006] and Agirrezabal et al. [2012b]. Each (automatically divided) syllable is linked to each of the tokens in the poem, uniquely identified by the `@id` attribute of each word form in the NAF document as can be seen in listing III.4.

When manually adding the metrical information about each line in a poem, this was done according to my own intuition by reading it aloud. But, if the poem had a sung version, I marked the stresses according to the respective song. From the selected 54 poems, 16 were tagged according to their sung version and 38 by reciting them.

To conclude, regarding the Basque corpus, there currently are two sub-corpora. One of them contains poems which have been marked according to their **recited** version and in the other one, the stresses have been marked according to a well-known song (**sung** version).

III.5 Summary of annotated corpora

To summarize, I have presented three corpora that will be used to train and test poetry scansion models. Some of them were previously annotated [Tucker, 2011, Navarro-Colorado et al., 2016] and in the case of the Basque language, I have performed annotations manually. When performing the manual annotation, some poems were tagged according to their sung version and some others according to the recited version. The sub-corpus of recited poems is expected to be more regular.

A general overview of the data can be observed in table III.4, where we can see the syllable, word and line frequencies of the three corpora (English, Spanish and Basque (recited)).

III.6 Other corpora

Several poems from Project Gutenberg[19] [Hart, 1971] were downloaded for training, testing and evaluating some of the prosodic and semantic models.

A dump from Wikipedia[20] was also used in order to train some semantic models.

[19] `http://www.gutenberg.org`

[20] dated in July-07-2014

	English	Spanish	Basque
No. syllables	10988	24524	20585
No. distinct syllables	2283	1041	920
No. words	8802	13566	7866
No. distinct words	2422	3633	4278
No. lines	1093	1898	1963

Table III.4: Word, syllable and line counts for each corpus.

The pronunciation dictionaries NETtalk [Sejnowski and Rosenberg, 1987] and CMU [Weide, 1998] were used, both of which list the pronunciations of the words, the number of syllables they contain, as well as indications of primary and secondary stress location. Each employs a slightly different notation, but they are, in general, quite similar in content as they both mark three levels of stress and show pronunciation:

```
NETTALK format:
@bdIkeS|n     '_'_     S4     abdication     0     (N)

CMU format:
INSPIRATION    IH2 N S P ER0 EY1 SH AH0 N
```

Finally, the Wall Street Journal section of the Penn Treebank [Marcus et al., 1993] was used to train a part-of-speech tagger, the role of which is described below.

CHAPTER IV

NLP techniques for scansion

Automatic scansion can be seen as a prediction problem, where getting each of the words in a poem we must draw conclusions about the accent that they are going to take. This prediction can be made in two different ways:

- following some **rules** that guide the marking, which are made by experts
- **learning from patterns in the observed data** and drawing conclusions from them, expecting that they will be representative.

In the next sections I describe the techniques used in the experimental work. First the rules and implementation of a rule-based approach are explored, and then I deepen in the data-driven systems. Data-driven methods are supervised if tagged information is included in the data. If tagged data is not available or it is not used, the methods are considered unsupervised. Supervised systems are divided in three main subgroups: Greedy predictors—i.e. the ones that classify each instance independently from the others—, structured predictors— which find an optimal resulting sequence— and neural networks (they work on both structured and unstructured output).

IV.1 Rule-based scansion

As other rule-based works reviewed in section II.2 [Gervas, 2000, Hartman, 2005, Bobenhausen, 2011, Navarro-Colorado, 2015, Delmonte, 2016], a rule-based method for poetry scansion was developed, presented in Agirrezabal

et al. [2013c, 2016b]. This program is released under the GNU GPL license and is available on Github.[1]

For this rule-based analyzer, a rather conservative approach was chosen, and one which also lends itself to a fairly mechanical, linguistic rule-based implementation. The system, which distinguishes three levels of stress internally, marks each line with a stress pattern and attempts to analyze the predominant meter used in a poem.

IV.1.1 Method

The tool is constructed around a number of guidelines for scansion developed by Peter L. Groves [Groves, 1998]. It consists of three main components:

(a) A simple implementation of Groves' rules of scansion —mainly a collection of POS-based stress-assignment rules.

(b) A pronunciation lexicon together with an out-of-vocabulary word guesser.

(c) A 'plausible foot division' system.

(a) Groves' rules

These rules try to assign stress levels so that, as far as possible, this becomes an objective process driven by lexicon and syntax, not dependent on more elusive concepts of the poem such as meaning and intent. The rules assign stress as follows:

1. Primarily stressed syllables of content words (nouns, verbs, adjectives, and adverbs) receive **primary stress**.

2. Secondarily stressed syllables in polysyllabic content words, primarily stressed syllables in polysyllabic function words (auxiliaries, conjunctions, pronouns, and prepositions) and secondarily stressed syllables in compound words get **secondary stress**.

3. Unstressed syllables of polysyllabic words and monosyllabic function words are **unstressed**.

In section IV.1.4 a more elaborate example is presented to illustrate how Groves' rules are implemented.

[1] `http://github.com/manexagirrezabal/zeuscansion`

(b) Pronunciation lexicon and out-of-vocabulary word-stress guesser

To calculate the lexical stress of words necessary for Groves' rules, the dictionaries mentioned in section III.6 are used: The CMU pronunciation dictionary [Weide, 1998] and NETtalk [Sejnowski and Rosenberg, 1987]. The system first attempts to locate the stress pattern in the smaller NETtalk dictionary (20,000 words) and then falls back to using CMU (125,000 words) if the word is missing in NETtalk. The merged lexicon, where NETtalk pronunciations are given priority, contains about 133,000 words.

In the event that a word is found in neither the NETtalk lexicon nor the CMU dictionary, the stress pattern of the word is guessed using a system that is based on finite-state technology (FST), which relies on the hypothesis that similarly spelled words have the same stress pattern.

(c) Foot division system

The final subtask is to divide a line's stress pattern into feet, for which a scoring system is used. The goal is to return the meter that the whole poem follows and to that end, the average stress value of each syllable position is calculated for the whole poem. The scoring system is used to resolve ambiguity cases and gives priority to triple meters over duple meters. More details are given below.

IV.1.2 General design

The structure of the system is divided into the subtasks shown in figure IV.1. It starts with preprocessing and tokenization, after which words are part-of-speech tagged. Following that, the lexical stress pattern for each word is found, guessing the stress patterns for any words not present in the dictionary. After these preliminaries, Groves' scansion rules are applied to know the prosodic stress and some cleanup of the result is done. Finally, *ZeuScansion* calculates the average line stress pattern, which later is divided into feet.

The toolchain itself is implemented as a chain of finite-state transducers, each of them written using the *foma*[2] toolkit [Hulden, 2009], save for the part-of-speech tagger which is a Hidden Markov Model (HMM) implementation [Halácsy et al., 2007]. The programming language *Perl* is used as a glue language to communicate between the components.

[2]https://foma.googlecode.com

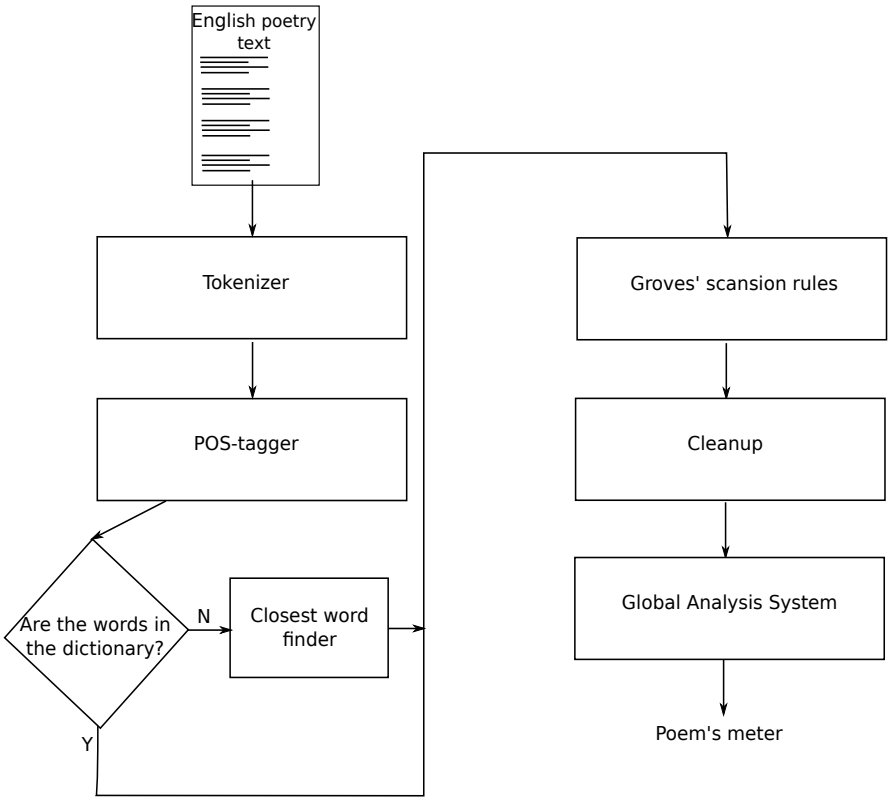

Figure IV.1: Structure of ZeuScansion

IV.1.3 *Part-of-speech* tagging and lexical stress assignment

After tokenization,[3] the part-of-speech (POS) tags of the words of the poem are obtained. For the POS-tagger, the software Hunpos[4] [Halácsy et al., 2007] was used trained with the Wall Street Journal English corpus [Marcus et al., 1993]. While other more general corpora might be more suitable for this task, the only need is to distinguish between function and non-function words, and thus performance differences are slight between tagger implementations. The evaluation of the HMM-based tagger and the function word/content word classifier can be found in the table IV.1. The evaluation environment from Tjong Kim Sang and Buchholz [2000][5] was used.

Once the first process is completed, the system starts applying Groves'

[3]The tokenizer's code can be found in https://code.google.com/p/foma/wiki/FAQ
[4]https://hunpos.googlecode.com
[5]`http://www.cnts.ua.ac.be/conll2000/chunking/`

Table IV.1: POS-tagger and FW/CW classifier evaluation in the CoNLL-2000 dataset.

Tagger	Precision	Recall	F1-Score
POS-tagger	94.97	92.85	93.61
FW/CW predictor	99.61	99.61	99.61

rules. This process is encoded as finite-state transducers. To apply the rules, however, the stress pattern of each word is supposed to be known. Here, as mentioned above, the system resorts to a heuristic for assigning lexical stress to out-of-vocabulary words.

As previously mentioned, the strategy used to analyze such words was to find a 'close' neighboring word in the dictionary, relying on an intuition that words that differ very little in spelling from the sought-after word are also likely pronounced the same way, or, at the very least, exhibit the same stress pattern.

In order to find the so-called 'closest word' in the dictionary, a cascade of finite-state transducers from the existing dictionaries is built in such a way that, given an input word, it will output the most similar word, according to spelling, using a metric of word distances that have been calculated for the purpose. These transducers will perform small specific changes (substitution, insertion, and deletion) in the input word, such as:

- Change one vowel
- Change one consonant
- Change two vowels
- Change one vowel and one consonant
- Change two consonants

Before performing any of these changes, the unknown word is divided into two parts, where the second part represents roughly the last syllable. Then, the aforementioned changes are performed in each part of the word. If, when performing any one of those changes, there is an existing word, the system will return that word and not proceed with the other changes. For example, in the following line from Shakespeare's *Romeo and Juliet*:

*And **usest** none in that true use indeed*

there is the word **usest**, which does not appear in the dictionaries (the archaic second-person singular simple present form of the verb **use**). The process of the closest word finder would begin with the word splitter, which would return "u.sest". Then, it would map this word to all possible words produced by changing just one vowel at the beginning, at the end, or changing one consonant. In this example case, after performing some of these changes the closest match will be found according to the scheme above: **wisest** and assume that its lexical stress matches that of **usest**—this is found by changing one vowel and inserting a consonant at the beginning of the word.

These transducers should be correctly ordered—an earlier transducer in the cascade will have priority over later ones. In the cascade, the dictionaries are also included as the very first mapping. If the word is not found in the dictionary, subsequent transducers perform the various mappings, filtering their outputs in such a way as to be constrained against possible words in the dictionary. The actual order in the cascade was determined based on the precision achieved. Cross-validation with against the NETtalk dictionary was used to calculate this precision against each ordering.

To illustrate this ordering, consider a pair of transducers, one performing just one vowel change and the other changing only one consonant. If the first transducer can guess the correct word in, say, 90% of the cases and the other one in 10% of the cases, the vowel transducer will be ordered first in the cascade, and the consonant transducer second. The final order of the transducers in *ZeuScansion* is:

1. Pronunciation dictionary.
2. Change one vowel at the left part.
3. Change one consonant at the left part.
4. Change two vowels at the left part.
5. Change one vowel and one consonant at the left part.
6. Change two consonants at the left part.
7. Change one vowel at the right part
8. Change one consonant at the right part.
9. Change two vowels at the right part.

10. Change one vowel and one consonant at the right part.

11. Change two consonants at the right part.

Listing IV.1: Foma code of the transducer that finds similarly spelled words.

```
%DICT: Transducer that accepts words if they are in the
    dictionary
%BEFORECHANGE: Adds vertical bar to mark the last syllable
%AFTERCHANGE: Removes the vertical bar
%VowChaBEF: Change one vowel if it is before the mark
%VowChaAFT: Change one vowel if it is after the mark
%ConChaBEF: Change one consonant if it is before the mark
%...
%VowChaBEF .o. VowChaBEF: Change two vowels before the mark



foma> regex DICT ;

foma> regex [BEFORECHANGE .o. VowChaBEF .o. AFTERCHANGE];
foma> regex [BEFORECHANGE .o. ConChaBEF .o. AFTERCHANGE];

foma> regex [BEFORECHANGE .o. VowChaBEF .o. VowChaBEF .o.
    AFTERCHANGE];

foma> ...

foma> regex [BEFORECHANGE .o. [[VowChaAFT .o. ConChaAFT] |
    [ConChaAFT .o. VowChaAFT]] .o. AFTERCHANGE];
foma> regex [BEFORECHANGE .o. ConChaAFT .o. ConChaAFT .o.
    AFTERCHANGE];

foma> save stack close-word-finder.fst

bash-user$ flookup -a close-word-finder.fst
```

IV.1.4 Groves' rules

Once we have obtained the lexical stress for each word, Groves' rules are applied using a finite-state transducer built from replacement rules [Beesley and Karttunen, 2003] that encodes each step in the rules.

Groves' rules dictate that the primarily stressed syllable in content words will maintain primary stress. In polysyllabic function words, the syllable carrying primary lexical stress will be assigned secondary stress. Secondary

stresses in polysyllablic content words will maintain secondary stress. All other syllables will be unstressed.

The input for these transducers is a string with this structure: "`word+POS`". The output will be the stress-pattern of the word after applying Groves' rules, written like: "`word+stress+POS`". Let's consider a line from the poem *The song of Hiawatha*:

changed them thus ***because*** *they mocked you*

Analyzing the word *because*: the input for the transducer that encodes Groves' rules would be "`because+IN`". The lexical resources would locate the word in the dictionary and this would return that the second syllable carries primary stress and that the first syllable is unstressed. After applying the prosodic stress rules, the system would return that the second syllable should receive secondary stress (instead of the original primary) as the input word is a polysyllabic function word. Hence, the output of the transducer in this case is "`because+x\+IN`".

The last step is to remove all the material not strictly required for working with stress patterns. In the cleanup process, a transducer removes everything before the first + character and everything after the second + character. It then removes all the + characters, so that the only result is the bare stress structure of the input word.

$$\texttt{because+x\+IN} \rightarrow \texttt{x\}$$

IV.1.5 Global analysis

After the stress rules have been applied and stressed syllables of each line are known, the meter inference process starts. To this end, the entire poem's average stress structure is calculated. This is encoded by a vector of syllable positions, whose value increments depending on the syllable's stress in each line. The pseudocode of the average stress calculator is as follows:

Listing IV.2: Pseudocode of the average stress calculator.

```
vector[1..nsylls]=0
foreach line (1..nlines) {
   foreach syllable (1..nsylls) {
      if stress(syllable) == /
          vector[syllable] = vector[syllable] + 2
      if stress(syllable) == \
          vector[syllable] = vector[syllable] + 1
   }
}
```

This process is illustrated with the poem No. 11 from "*The song of Hiawatha*" by Henry Wadsworth Longfellow:

(1) *Barred with streaks of red and yellow*
(2) *Streaks of blue and bright vermilion*
(3) *Shone the face of Pau-Puk-Keewis*
(4) *From his forehead fell his tresses*
(5) *Smooth and parted like a woman's*
(6) *Shining bright with oil and plaited*
(7) *Hung with braids of scented grasses*
(8) *As among the guests assembled*
(9) *To the sound of flutes and singing*
(10) *To the sound of drums and voices*
(11) *Rose the handsome Pau-Puk-Keewis*
(12) *And began his mystic dances*

The stress values for each line are the following:

(1) `/x\x/x/\`
(2) `\x/x/x/x`
(3) `/x/x?`
(4) `xx/\/x\x`
(5) `/x\xxx\x`
(6) `\x/x/x\x`
(7) `/x\x\x\x`
(8) `/x\x\x\x`
(9) `xx/x\x\x`
(10) `xx/x\x\x`
(11) `/x/x?`
(12) `xx\x/x\x`

Syllable	1	2	3	4	5	6	7	8
Count (stressed)	14	0	19	1	14	0	12	1
Normalized	0.74	0	1	0.05	0.74	0	0.63	0.05
Stress	/	x	/	x	/	x	/	x

Table IV.2: Each syllable's average stress value calculation.

The numbers in table IV.2 represent each syllable's stress over the entire poem. The numbers in the second row show the total sum of each syllable

position's stress value, calculated following the method listing IV.2. The values from the third row are the values from the previous row, normalized against the highest value. In figures IV.2 and IV.3 a graphical representation of these last numbers is shown based on an analysis of Shakespeare's *Sonnets* and Longfellow's *The song of Hiawatha*. As for the meter inference process two levels of stress are needed, a cutoff value of 0.5 is used: if the normalized average stress for a syllable is greater than this, it is assigned the label 'stressed' and otherwise 'unstressed'.

For this calculation, it is assumed that all the lines contain the same number of syllables. This naturally leads to difficulties with certain works with differing syllable counts per line (such as *Phantasmagoria* and other poems by Lewis Carroll). The interesting problems surrounding proper normalization and treatment of mixed-line poems is set aside to future work.

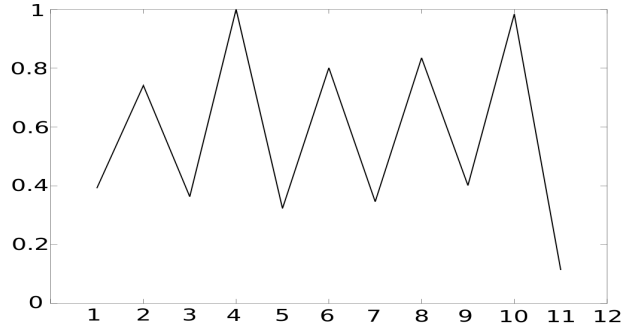

Figure IV.2: Average stress for each syllable in Shakespeare's Sonnets. Because of the poems' regular iambic structure, it is remarkable the rising patterns in the 1-2, 3-4, 5-6, 7-8, 9-10 syllables.

After the above steps, the system attempts to divide the average stress pattern into feet with the goal of producing a global analysis of the poem. In the previous example (`/x/x/x/x`), the optimal meter to assign is trochaic tetrameter, a sequence of four trochees (`[/x][/x][/x][/x]`). In the case that the assigned meter was, e.g., iambic, the first and last stresses would not fit (`/[x/][x/][x/]x`). In other cases foot-division can be more challenging. Consider, for instance, the analysis of a line containing 12 syllables:

`/xx/xx/xx/xx`

This verse could be analyzed as consisting mainly of (1) dactyls [`/xx`], (2) anapests [`xx/`], (3) trochees [`/x`] and (4) iambs [`x/`]. Dactylic and trochaic

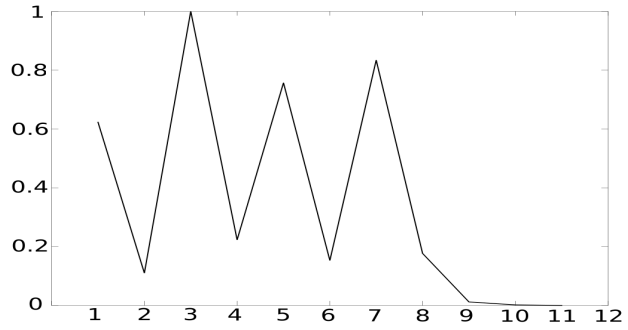

Figure IV.3: Average stress for each syllable in Henry Wadsworth Longfellow's *The Song of Hiawatha*. Because of the poems' trochaic nature, it is worth mentioning the dropping patterns in the 1-2, 3-4, 5-6 and 7-8 syllables.

Foot	Pattern	Nº matches	Score
Dactyl	`/xx`	4	6
Anapest	`xx/`	3	4.5
Trochee	`/x`	4	4
Iamb	`x/`	3	3

Table IV.3: Hypothetical feet for the meter in the example.

patterns appear four times in the line, however, anapestic or iambic patterns three times. By choosing the most frequent, ZeuScansion has to decide which pattern is the poem following, as both the dactyls and trochees appear four times. For such cases, a scoring system is used for selecting the appropriate pattern: a weight of 1.0 is given for hypothetical disyllabic patterns, and a weight of 1.5 for trisyllabic ones. In this example, this would yield the judgment that the structure is dactilyc tetrameter (1.5×4 matches $= 6$). This example is illustrated graphically in table IV.3.

IV.2 Supervised learning

As we talk about supervised learning paradigms, the models need data to learn patterns from. This section is divided in four subsections. In the first part I show the feature templates used for learning and in the next three

subsections, I discuss different sets of models: greedy prediction, structured prediction and neural networks.

IV.2.1 Modeling and features

When working with statistical learning methods, the representation of the data that is going to be learnt is important. These representations vary from problem to problem, but the process of extracting this information is often similar.

The current task is to perform scansion, given a poem. Hence, the goal is to assign stress values to syllables from a poem. The problem of assigning stress to each syllable can be then seen as

$$f(syll_1, syll_2, ..., syll_N) = (stress_1, stress_2, ..., stress_N)$$

The function f will get a sequence of syllables as input and will return a sequence of N values ($\{0,1\}^N$ or $\{x,/\}^N$), defining each element, $stress_i$, as a stressed or unstressed syllable.

A simple method for calculating these stresses, would be to calculate the stress value for each syllable by simply checking the conditional probability of a stress value, given a syllable. This is what one of the baselines, Naive Bayes, performs.

$$x_1 = P(stress_i = /|syllable = syll_i)$$
$$x_2 = P(stress_i = x|syllable = syll_i)$$

Then, for each syllable in a sentence, the highest probability from x_1 and x_2 is picked.

Current algorithms used in machine learning rely on extended features that give more information about the data in question. For example, the lexical stress of a syllable is informative for this task, as it was used for *ZeuScansion*. In the rule-based system, it was also seen that knowing the POS-tag of the current word is a good hint, as content words receive generally more stress than function words.

Below, the set of feature templates used in the machine learning based scansion systems are shown, which include basic and additional features:

(a) Basic feature templates. They are (almost) language agnostic:

- Syllable number within the word (SNOW): This specifies the current syllable position within the word. E.g., for the word *ha-zel*, whose lexical stress is `/x`, the specification of the current syllable gives information about the lexical stress of the current syllable.
- Syllable number within the line (SNL): This feature helps to model the sequence in many types of specially metered lines. It can resolve potentially ambiguous cases, e.g. the word re-cord. It's lexical stress can be both `x/` and `/x`, if it is a verb or a noun, respectively. Without knowing the part-of-speech tag, if we know that this word appears in the last two positions of a trochaic poem, we can ensure that it's lexical stress will be `/x`.
- Number of syllables in the line (NSL): The combination of this feature and the previous one helps in identifying the syllables at the end of a line which are usually more regular because of rhyme patterns.
- Syllable phonological weight (SWEIGHT): Around a third of the world's stress systems are weight sensitive [Ryan, 2016]. This feature relies on the cross-linguistic generalization that states that heavier syllables[6] attract stress and lighter syllables are commonly unstressed [Hayes, 1995, Gordon, 2002, 2004]. Because of that, this information could be useful in scansion systems, as reflected in the poem "*To Autumn*" by John Keats, "*to* ***swell*** *the* ***gourd*** *and* ***plump*** *the* ***hazel*** ***shells***".[7]
- The last 5 characters of the word (last character, last two characters, last three characters, last four characters, and last five characters) (LC1...LC5): As primary stress of the words in English is usually concentrated in the last syllables of the word (roughly the last three syllables) [Hayes, 1995, p. 50], the last characters were expected to be informative. Although it could be better to use the last characters of the syllable, as it was done in Estes and Hench [2016], in this work the last characters of the word are used, so as to be more agnostic about the language in question.
- Word length (WLEN): This was expected to be an informative feature.

(b) Additional feature templates:

[6] Heavy syllable: The syllable has a coda or ends in a tense vowel.

[7] In this example an underlined syllable represents a heavy syllable.

- Word: As the main basic units of the text, words are used as features.
- Syllable: Some syllables are almost always stressed, which could help in the inference of stress patterns. For example, in Shakespeare's Sonnets, the syllable "sire" is used 10 times and in all of them it appears as stressed.
- POS-tag: The part of speech is a key element to decide whether a word is a content word or function word, which affects the stress in many syllables, as in the following excerpt from *The voice* by Thomas Hardy: "*call to me call to me*", both the verb *call* and the pronoun *me* are stressed, but the pronoun loses the prominence when read aloud because it is not a content word. Previous works on poetry analysis, such as Groves [1998], rely on this information.
- Lexical stress (LS): Knowing the lexical stress sequence in a phrase is an important hint for deducing the rhythmic pattern of a line of poetry. The lexical stress of the current word is included. This lexical stress is calculated by using the NETTalk dictionary [Sejnowski and Rosenberg, 1987] and when treating out-of-vocabulary words, their stress is calculated using the SVM implementation given in Agirrezabal et al. [2014b].

These last four features are extended to include their context as well. For each syllable in the data, the current syllable, $syllable[t]$, is taken into account together with its previous and next 10 syllables, syllable[t$\pm$10]. In the case of words, part of speech tags, and lexical stresses the $\pm$ 5 surrounding elements are included. This was done with the intuition that each word could have approximately two syllables.

To conclude about feature information, an example from a line of poetry will be presented together with the feature template values. Let us consider a line from *Scrambled Eggs Super!*, by Dr. Seuss [Seuss, 1953]:

I really cooked something worth talking about

The feature templates per syllable of this line are shown in table IV.4. Some attributes like *Syllable*, *Word*, *Lexical stress*, or *POS-tag*, are extended with the surrounding elements as learning features to model the context. As mentioned above, the next and previous five elements are used in the case of words, lexical stresses and POS-tags. The context of syllables is modeled with ten surrounding elements.

Syllable	I	real	ly	cooked	some	thing	worth	talk	ing	a	bout
Word	I	really	really	cooked	something	something	worth	talking	talking	about	about
Lexical stress	'	'_	'_	M'_	'_	'_	'	M'_	M'_	_'	_'
POS-tag	PRP	RB	RB	VBN	NN	NN	IN	VBG	VBG	IN	IN
Syllable no. word	0	0	1	0	0	1	0	0	1	0	1
Syllable no. line	0	1	2	3	4	5	6	7	8	9	10
No. syllables line	11	11	11	11	11	11	11	11	11	11	11
Word length	1	6	6	6	9	9	5	7	7	5	5
Syllable weight	0	1	0	1	0	1	1	1	1	0	1
Last char	I	y	y	d	g	g	h	g	g	t	t
Last 2 chars	#i	ly	ly	ed	ng	ng	th	ng	ng	ut	ut
Last 3 chars	##i	lly	lly	ked	ing	ing	rth	ing	ing	out	out
Last 4 chars	###i	ally	ally	oked	hing	hing	orth	king	king	bout	bout
Last 5 chars	####i	eally	eally	ooked	thing	thing	worth	lking	lking	about	about
Class	x	/	x	x	/	x	x	/	x	x	/

Table IV.4: Feature templates for a line from "Scrambled Eggs Super!" by Dr. Seuss. Syllables are extended with the ± 10 elements and the next three elements (words, lexical stresses and POS-tags) with the ± 5 elements.

IV.2.2 Single/greedy prediction

In this subsection, greedy predictors are explored, which are the ones that classify each instance independently and do not find the optimal resulting sequence.

IV.2.2.1 Naive Bayes

Naive Bayes methods are a set of supervised algorithms [John and Langley, 1995] based on the well-known Bayes' theorem with the *naive* assumption of independence between every pair of features. Under this assumption, calculating the probability of a class y generating a set of features is performed by multiplying each of the conditional probabilities. By applying Bayes' theorem we have that:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(x_1, x_2, ..., x_n|y) \times P(y)}{P(x_1, x_2, ..., x_n)} \tag{IV.1}$$

and as all the features are assumed to be independent, the probability of an instance based on its attributes, becomes simply

$$P(y|x_1, x_2, ..., x_n) = \frac{P(x_1|y) \times P(x_2|y) \times \ldots \times P(x_n|y) \times P(y)}{P(x_1, x_2, ..., x_n)} \tag{IV.2}$$

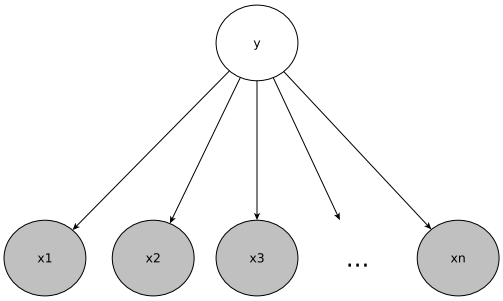

Figure IV.4: The output variable y is dependent on a set of features $x_1, x_2, \ldots x_N$ and these features are assumed to be independent.

This classifier has been used in several tasks [Manning and Schütze, 1999], such as Word Sense Disambiguation [Gale et al., 1992]. Nowadays, it is used as a popular method as baseline of different natural language processing problems, although its performance can achieve comparable results as other more complex algorithms [Pang et al., 2002].

In this work, I have used an implementation from the Weka software suite [Hall et al., 2009, Witten et al., 2016].

IV.2.2.2 Perceptron

The perceptron [Rosenblatt, 1958] is a very simple algorithm for binary classification and is inspired by biological neurons. Its structure is similar to some specific parts of Neural Networks. The perceptron has a set of inputs and a binary output, as it can be seen in figure IV.5. The output of this unit can be calculated by the dot product of the input vector with the weights vector and then later applying the Heaviside step function[8] as activation:

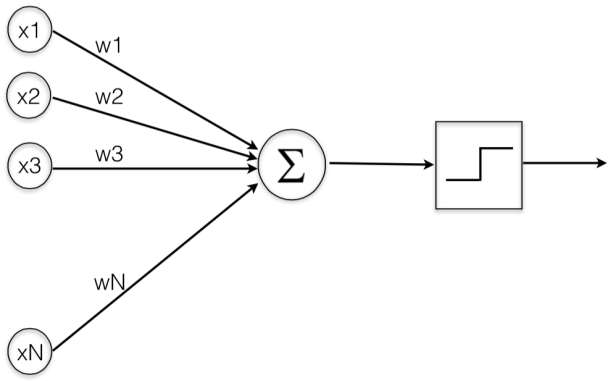

Figure IV.5: Structure of a perceptron.

$$\begin{aligned} out &= \sum_{i=0}^{n} x_i w_i \\ y &= H(out) \end{aligned} \tag{IV.3}$$

The idea behind the perceptron was formulated by McCulloch and Pitts [1943], and Rosenblatt [1958] presented the Perceptron including a simple algorithm to learn its weights.

Although at the beginning it seemed to be promising, Minsky and Papert showed that not all the problems can be solved using a Perceptron, such as, the XOR problem [Minsky and Papert, 1969]. The algorithm, shown

[8]The Heaviside step function is a simple function that for any negative number returns 0, otherwise it returns 1.

in listing IV.3, will get a solution only if the two sets of data are linearly separable. This is the basic idea of a Perceptron, with its simple learning algorithm. When this learning method is used, the Perceptron is referred as Vanilla Perceptron.

Listing IV.3: The Perceptron learning algorithm.

```
W = [0.0, 0.0, 0.0, 0.0, 0.0] #Weight vector
X = [[1,1,1,1,1],[0,0,0,0.5,0],...] #Input instances
Y = (0, 1, 1, 0, ... 1) #Gold labels
for i in (1..n):
  prediction = classify(X,W,i)
  if prediction == Y[i]:
    do nothing
  else:
   if prediction == 1:
      w = w - X[i]
   else:
      w = w + X[i]
```

The Perceptron has two main disadvantages. The first is that it needs linearly separable data. The second problem is that the order of instances can affect to the resulting separating hyperplane. For example, let us suppose to have a corpus composed by 1000 instances and a model that is able to learn a very good separating hyperplane in the first 950 instances. If the last 50 instances are outliers, the good hyperplane learned until this point will be completely lost. To solve these two problems, a commonly used solution is to average the way in which each instance affects to the hyperplane. This is performed in the Averaged Perceptron [Freund and Schapire, 1999, Daumé III, 2012].

The Perceptron classifier has been extensively used in NLP, e.g. in Collins and Roark [2004], Shen and Joshi [2005], Carreras [2005], Sak et al. [2007], Alegría et al. [2008], Jiang et al. [2008], Arrieta et al. [2014], Otegi et al. [2016].

In this work, I have used the Averaged Perceptron. A publicly available implementation of this Perceptron can be found in this repository at Bitbucket.[9]

IV.2.2.3 Support Vector Machines

Support Vector Machines [Cortes and Vapnik, 1995, Hsu et al., 2003] are supervised learning models for performing classification or regression analy-

[9] https://bitbucket.org/mhulden/pyperceptron

sis. It performs non-probabilistic linear binary classification. This technique represents each instance of a problem in an n-dimensional hyperplane and it tries to find the separating $(n-1)$-dimensional hyperplane between the two classes. Theoretically, if data points are linearly separable, there are infinite separating hyperplanes, but the SVM will maximize the margin between the two sets of instances. As it was said before, each instance is represented in an n-dimensional space, representation that can be found with the following formula

$$y(x) = \Phi(x) + b \tag{IV.4}$$

where $\Phi(x)$ is a feature-mapping function that taking x as input argument, returns a point in the n-dimensional space. In such space there will be two separating hyperplanes

$$[\Phi(x) + b = -1] \tag{IV.5}$$
$$[\Phi(x) + b = 1] \tag{IV.6}$$

that will divide it in two subspaces. These separating hyperplanes (equations IV.5 and IV.6) can be easily appreciated in figure IV.6. Following the linear separability assumption, the classification of new instances is straightforward, then.

$$f(x) = \begin{cases} class = 1, & \Phi(x) + b > 0 \\ class = 0, & \Phi(x) + b < 0 \end{cases} \tag{IV.7}$$

Unfortunately, linear separation is not always possible, and in that case, a solution is to use a soft-margin, so that some errors are accepted. In order to use soft-margin, the maximizing formula must be extended by using the hinge-loss function. The hinge loss is an approximation to the misclassification error, which means that wrongly classified instances are penalized based on their distance from the separating hyperplane. Another solution for the non-separability problem is to use the so-called kernel trick. In this way, the original n-dimensional instances are mapped in a higher dimensional space and linear separability is often achieved. Some of the commonly used kernel functions are polynomial, Radial Basis Function (RBF) and sigmoid.

Support Vector Machines work well when the feature set is large, e.g. when bag-of-word models are used. Because of their appropriateness, SVMs have been widely used in Natural Language Processing, for instance, in text

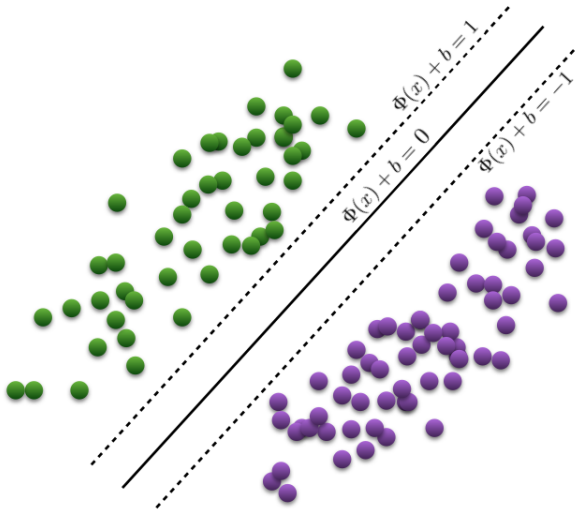

Figure IV.6: Support Vector Classification in a two-dimensional space.

categorization [Dumais et al., 1998], part-of-speech (POS) tagging [Nakagawa et al., 2001, Mohan et al., 2010], named-entity recognition (NER) [Isozaki and Kazawa, 2002, Kazama et al., 2002], chunking [Kudo and Matsumoto, 2001], dependency analysis [Yamada and Matsumoto, 2003] or information extraction [Li et al., 2005].

In this work the LibLinear and LibSVM packages [Fan et al., 2008, Chang and Lin, 2011] included in the Weka software suite [Hall et al., 2009, Witten et al., 2016] have been used.

IV.2.3 Structured prediction

We now get involved in structured prediction methods, where given an input sequence an output sequence must be produced. This output sequence will be the optimal result from a set of possible outputs. The reader can refer to subsection II.3.2 to see the advantages of doing predictions jointly instead of doing them independently.

IV.2.3.1 Hidden Markov Models

Hidden Markov Models (HMM) are simple and useful models [Rabiner, 1989, Bengio, 1999] to solve sequence-to-sequence problems. Under Hidden Markov Models, output tags are considered states and this output state sequence is modeled using automata.

When there are dependencies between outputs, or states, a problem can be represented as a Markov process, which consider a state dependent only on its previous state (Markov assumption) with a probability p. Such process will consist of

- M states —the number of possible outputs.
- A transition matrix of size $M \times M$ —the transition probabilities between different states.
- M initial probabilities —probability of starting at each state.

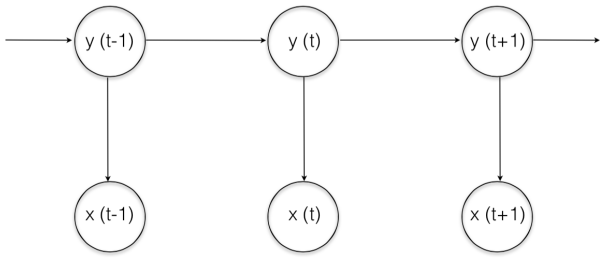

Figure IV.7: Temporal classification with Hidden Markov Models.

Then, the output sequence is represented as a Markov process, but how does this process relate with the elements that are observed—i.e. data? As it is represented in figure IV.7, the upper part shows the interaction between outputs at each timestep ($[y_{t-1} \rightarrow y_t]$ and $[y_t \rightarrow y_{t+1}]$) and these probabilities will be got from the transition matrix. Under the outputs, the observed data can be seen ($\dots x_{t-1}, x_t, x_{t+1} \dots$).

In order to calculate the probability of associating a state sequence with an input observed sequence, the probability between the data and the states at each timestep must be considered. For this calculation, an additional table is needed, that includes the so-called emission probabilities, or the probability of generating, i.e. emitting, an instance given a specific state ($P(x_t|y_t)$). With these elements, a complete sequences probability can be calculated by

using the initial probabilities, emission probabilities and transition probabilities.[10]

$$P(y_{1..N}, x_{1..N}) = \prod_{t=0}^{seqlength} \underbrace{P(y_{t-1}|y_t)}_{Transition\_probability} \times \underbrace{P(x_t|y_t)}_{Emission\_probability} \quad \text{(IV.8)}$$

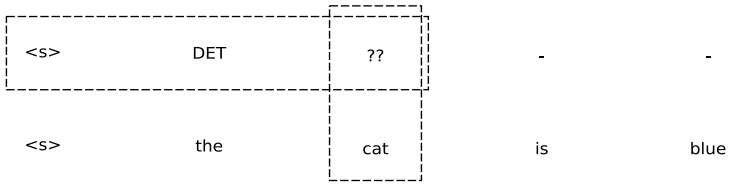

Figure IV.8: POS tagging using a second-order Hidden Markov Model.

Usually, in practical applications such as POS-tagging, higher order Hidden Markov Models are used, where the probability of a state is given by it's previous n elements (n-th order HMM). In figure IV.8, we can see which elements are considered in order to calculate the POS-tag of a word using a second order HMM. Summarizing, these are the assumptions made in HMMs:

1. The Markov assumption: The probability of a state is just dependent on the previous state (or previous n states in the case of n-th order models).

2. The stationarity assumption: The transition probabilities going from y_i to y_{i+1} in the model are always the same, independent of the moment in which it happens.

3. The observation independence assumption: There is no conditional dependency between the observations, i.e., they are supposed to be independent.

Being stated these assumptions, as stated by Rabiner [1989], these are the main problems of HMMs:

1. Decoding

[10] Initial probabilities are used when $t = 0$.

2. Inference

3. Parameter estimation

Decoding

This involves calculating the probability of an observed sequence taking as input the model and the sequence of hidden states. The forward procedure of the forward-backward algorithm (dynamic programming), solves this problem [Rabiner, 1989]. Using this procedure, the formula IV.8 is applied.

Inference

It is the optimal sequence given a model and an observed variable sequence. When a sequence of elements is given and the most likely hidden state sequence must be returned, according to a model, the maximum over all possible state sequences has to be calculated. This can be solved calculating the probabilities of all possible sequences and returning the maximum over all. The Viterbi algorithm [Viterbi, 1967, Forney, 1973] solves this problem efficiently.

Parameter estimation

The parameters of Hidden Markov Models (initial probabilities, emission probabilities and transition probabilities) are learned easily by using the maximum-likelihood estimation principle (MLE) as it can be seen in equations IV.9, IV.10 and IV.11:

$$P(y_t = a) = \frac{Count(y_t = a)}{Count(y_t)} \tag{IV.9}$$

$$P(y_{t-1} = a | y_t = b) = \frac{Count(y_t = b, y_{t-1} = a)}{Count(y_t = b)} \tag{IV.10}$$

$$P(x_t = s | y_t = a) = \frac{Count(y_t = a, x_t = s)}{Count(y_t = a)} \tag{IV.11}$$

The use of Hidden Markov Models is typical in Natural Language Processing as a simple solution to sequential tagging problems [Brants, 2000, Halácsy et al., 2007, Ponomareva et al., 2007], but also for speech recognition [Young and Young, 1993, Erro et al., 2010].

In this study, *Hunpos*, a publicly available implementation of HMMs [Halácsy et al., 2007] was used, whose code can be downloaded from Google Code.[11]

IV.2.3.2 Conditional Random Fields

A Conditional Random Field [Lafferty et al., 2001, Sutton and McCallum, 2011] is a statistical model that is used for structured prediction. As Logistic Regression models are to the Naive Bayes model, linear-chain CRFs are considered the discriminative counterpart of Hidden Markov Models [Sutton and McCallum, 2011, p. 19], as it can be seen in figure IV.9.

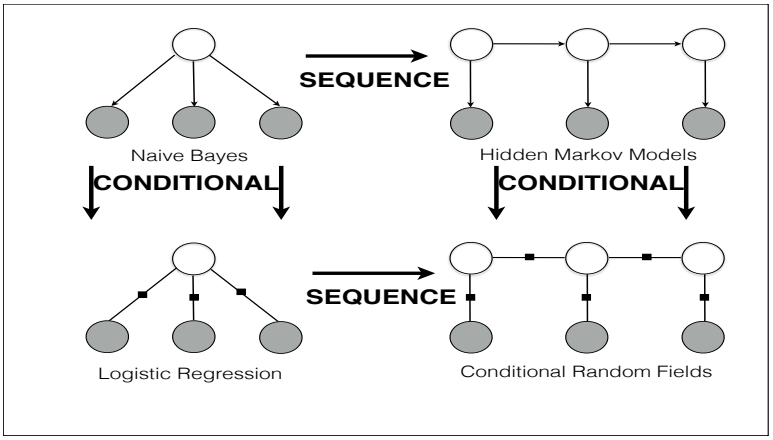

Figure IV.9: Conditional Random Fields are the conditional counterpart of Hidden Markov Models and the sequential version of the Logistic Regression model [Sutton and McCallum, 2011, p. 19].

CRFs model the conditional distribution $P(y|x)$ and not the joint $P(y, x)$ distribution, as the HMMs do. This makes it a much simpler and more efficient approach because it focuses on the differences among the instances of different classes. Focusing on the differences, the number of combinations of possible hidden/observed mappings is significantly lower. In HMMs generative learning is used for discrimination. When the dimensionality of x is very large or it has complex dependencies, constructing a probability distribution over it ($P(x)$) is difficult in the generative classifier.

[11] `http://hunpos.googlecode.com`

Because of the discriminative nature of CRFs, more features can be easily incorporated in the models. The addition of extended features for the representation of the input in CRFs is a crucial element for the success of these models. The features are represented as feature functions f_k, dependent on the current hidden state, previous hidden state and any element from the observed sequence. The x_t from equation IV.12 should be understood as any element from the observed sequence (x) [Sutton and McCallum, 2011, p. 23].

$$P(str_{1..N}|syll_{1..N}) = \frac{1}{Z(x)} \prod_{t=1}^{T} exp\left\{\sum_{k=1}^{K} \theta_k * f_k(y_t, y_{t-1}, x_t))\right\} \quad \text{(IV.12)}$$

$$Z(x) = \sum_{y} \prod_{t=1}^{T} exp\{\sum_{k=1}^{K} \theta_k * f_k(y_t, y_{t-1}, x_t))\} \quad \text{(IV.13)}$$

In the same way as in HMMs, parameter estimation is an important part of CRFs. This is done by Maximum Likelihood by defining a conditional log likelihood function

$$l(\theta) = \sum_{i=1}^{N} logp(y^{(i)}|x^{(i)}) \quad \text{(IV.14)}$$

where if we join equations IV.12 and IV.14 we will get the following expression.

$$l(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) - \sum_{i=1}^{N} logZ(x^{(i)}) \quad \text{(IV.15)}$$

A problem concerning Machine Learning, and thus, CRFs, is overfitting. So that to overcome it, different techniques are used, for example a regularization parameter. This parameter adds a penalization to all features so that to avoid some specific features to dominate without a motivated reason. A regularization parameter can be added in this log-likelihood formula.

Given the log-likelihood function, the goal now is to maximize it along its derivative (gradient). So that to do this, quasi-Newton methods such as the *Limited-memory BFGS* are used [Nocedal and Wright, 2006].

CRFs have been widely used in NLP, especially in Named Entity Recognition [McCallum and Li, 2003, Ponomareva et al., 2007] and also as a layer

of a Neural Network model, as in Lample et al. [2016], but also in other disciplines, such as in bioinformatics [Sato and Sakakibara, 2005].

In this work I have employed the publicly available CRF model implementation `CrfSuite`[12] [Okazaki, 2007], and in order to be usable from python, I have used `pyCRFsuite`[13], a wrapper for Python.

IV.2.4 Neural Networks / Deep Learning

A Perceptron can work well with linearly separable problems, such as the AND, OR or NOT logical operations. However, it fails to correctly perform in the case of the XOR operation [Minsky and Papert, 1969], as it was seen before. In order to solve this problem, Multilayer Perceptrons were proposed, which could work with the XOR operation and other non-linearly separable classification problems. These complex systems evolved until the current state-of-the-art Deep Learning methods [Graves, 2012].

IV.2.4.1 Multilayer Perceptron

The Multilayer Perceptron is a Feed-forward Neural Network [Minsky and Papert, 1969, Rumelhart et al., 1988a]. The name feed-forward comes from the fact that its connections do not cycle, they always go forward from the input through all mathematical operations until the desired output is reached. The network is composed of a set of nodes that perform a linear computational operation such as,

$$A = Wx + b \tag{IV.16}$$

where W represents a weight matrix, x is the input vector and b is a bias term. This calculation is identical to the one performed in the Perceptron save for the activation function, which is different.

The goal is to get a black box with a set of nodes performing simple computational operations and to solve (almost) any computational problem. The hard task is the learning process of the necessary weights (W).

A simple version of a feed-forward neural network is a single-layer perceptron network, which includes a set of input values and the result is a set of outputs. The special case of a network that consist of only one layer, a single output value and this value is calculated by a linear activation function

[12] `http://www.chokkan.org/software/crfsuite/`
[13] `https://github.com/jakevdp/pyCRFsuite`

—specifically, the Heaviside step function[14]— is the Perceptron. A single-layer neuron with a logistic function[15] for the activation is identical to the logistic regression model [Bishop, 2006].

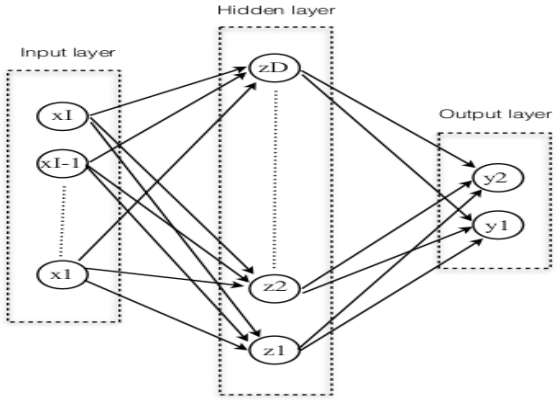

Figure IV.10: Example of a prototypical Neural Network Bishop [2006].

In more complex neural networks, such as the one showed in figure IV.10 nodes are organized in layers. The output values from nodes in layer i are the input values to the nodes in layer $i+1$. There is a set of weights that controls the relevance of the connection between nodes on subsequent layers. The outputs of neural networks are typically calculated by functions like the logistic function, instead of using a step function, as the derivative of the logistic function can be easily calculated. Feed-Forward Neural Networks are models that are able to find non-linear patterns from data. Weight learning is done by backpropagation [Rumelhart et al., 1985] [Bishop, 2006, p. 241-244], which is an efficient technique for the evaluation of the gradient of an error function $E(w)$ for a feed-forward neural network.

A neural network can be understood as a graph of nodes that perform a computation. Each of these nodes has a set of I inputs $z_i|i \in \{1..I\}$. These inputs are then multiplied by the nodes weights, $w_i|i \in \{1..I\}$ which results in

$$a_j = \sum_{i=1}^{I} w_{ji} z_i + w_{j0}^{(1)} \tag{IV.17}$$

[14]The Heaviside step function is a simple function that for any negative number returns 0, otherwise it returns 1.

[15]A similar function as the Heaviside step function, but easily differentiable.

and a function $h()$ is applied to this

$$z_j = h(a_j) \tag{IV.18}$$

where $h(x)$ is an activation function like the hyperbolic tangent (equation IV.19) or the sigmoid function (equation IV.20).

$$h(x) = tanh(x) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{IV.19}$$

$$h(x) = \frac{1}{1 + e^{-x}} \tag{IV.20}$$

After defining the mathematical operations that will be made in the network, we need to adjust the weights previously mentioned. These weights are learned by minimizing the error in the data $(x_1, x_2, \ldots x_N)$. The error of a neural network can be defined as a function that depends on the weights of the input nodes.

$$E(w) = \sum_{n=1}^{N} E_n(w) \tag{IV.21}$$

The gradient of the error function with respect to a weight w_{ji} is given by its derivative:

$$\frac{\partial E_n}{\partial w_{ji}} \tag{IV.22}$$

The weights are then updated based on these derivatives and a prespecified learning rate, by the application of the delta rule. The error can be minimized using gradient descent.

IV.2.4.2 Recurrent Neural Networks

Recurrent Neural Networks [Rumelhart et al., 1988a, Elman, 1990, Werbos, 1990] (RNN) are the generalization of Multilayer Perceptrons for sequential data. They can be useful for structured prediction problems, such as sequence labelling, where a string of outputs must be returned given an input sequence.

In Feed-forward Neural Networks like Multilayer Perceptrons there was not any cycle between nodes. All connections were done in a forward direction. In Recurrent Neural Networks, this limitation is relaxed and connections are allowed making these models much more powerful and expressive. In sequence-to-sequence problems, we are given an input sequence

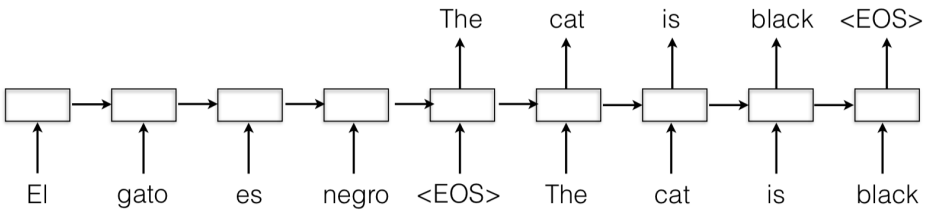

Figure IV.11: Example of a simple encoder-decoder machine translation system.

$x_1, x_2, \ldots, x_N$ and an output $y_1, y_2, \ldots, y_N$ must be produced, for which the following equation is iterated [Graves, 2012]:

$$h_t = f(W^{hx} x_t + W^{hh} h_{t-1}) \tag{IV.23}$$

$$y_t = g(W^{yh} h_t) \tag{IV.24}$$

In this formulation there are two important variables. The variable h_t (equation IV.23) represents the hidden representation of the memory at timestep t, which will include all the collapsed information after analyzing a sequence of elements $x_0 \ldots x_t$. This *memory* is represented as a real number vector and is calculated as a combination of the input element x_t, weighted with W^{hx}, and the hidden representation at the previous timestep h_{t-1}, weighted with W^{hh}. After calculating this, typically a non-linear activation function, such as the sigmoid or the hyperbolic tangent, is applied (the f function in the equation).

This memory h_t can be used for prediction at each timestep (IV.24), which will be multiplied by the hidden-to-output weights W^{yh} and after that an activation function g will be applied. Figure IV.12 shows an actual example of this type of models. This is appropriate in tasks where predictions must be made for each input element, such as, POS-tagging, named entity recognition, ... It was used in Xu et al. [2015].

Another typical model used for sequence modelling is the so-called encoder-decoder architecture. Figure IV.11 shows how these models are used for machine translation. It encodes a sequence input using a RNN into a fixed-sized vector, which is the dense representation of the input sequence x. This representation is mapped into a target sequence by using another RNN, which

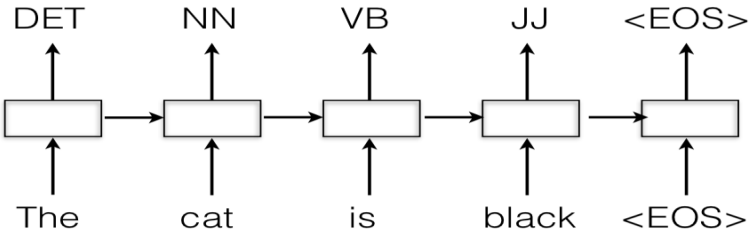

Figure IV.12: Example of the use of a RNN giving an output for each input, e.g., in a POS-tagger.

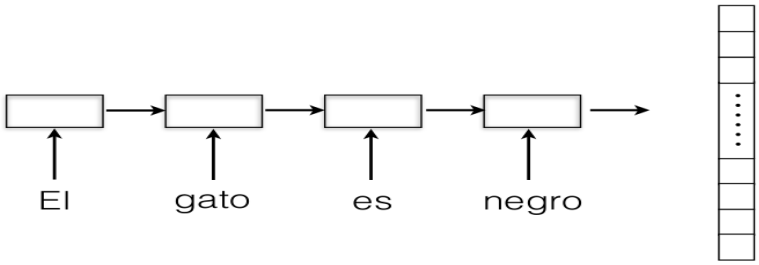

Figure IV.13: Example of an encoder, which encodes the input sequence in a vector.

will decode the dense representation into the desired output. When learning, error gradients are calculated based on the learning data and the produced output. These gradients are propagated through all connections to the input side [Werbos, 1990].

The advantage of this architecture is that there is no necessity of having an output for each input, as the input is first mapped into an internal representation which will generate the proper output. In order to do that, the RNN calculates the dense representation c of the input sequence x and it will start to generate a sequence y by making predictions:

$$\begin{aligned} h_{(t)} &= f(h_{(t-1)}, y_{t-1}, c) \\ P(y_t|y_{t-1}, y_{t-2}, \ldots, y_1, c) &= g(h_{(t)}, y_{t-1}, c) \end{aligned} \tag{IV.25}$$

Following the definition in equations IV.23 and IV.24, RNNs can be

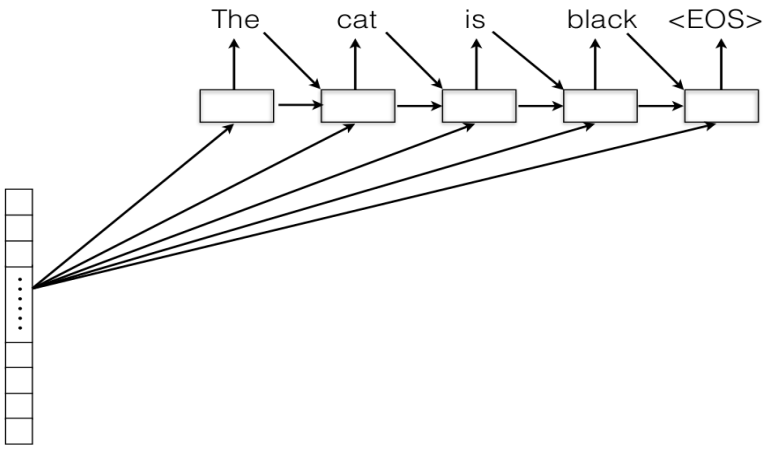

Figure IV.14: Illustration of a RNN that, getting the output of an encoder c, the current memory state and the previous outputs, produces an output maximizing its probability.

graphically represented as recursively, as in figure IV.15, and this can be unrolled for a finite sized input sequence (figure IV.16).

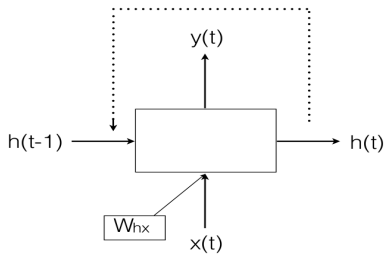

Figure IV.15: Traditional (recursive) representation of Recurrent Neural Networks.

When checking the unrolled representation of the RNN, it can be seen that the training procedure is the same as in other Neural Networks. It is done by backpropagating errors, and in the RNN literature this is referred to as *backpropagation through time* (BPTT) [Werbos, 1990].

This model, the Encoder-Decoder, was used in works like Sutskever et al. [2014], Cho et al. [2014], Bahdanau et al. [2014], Kann and Schütze [2016].

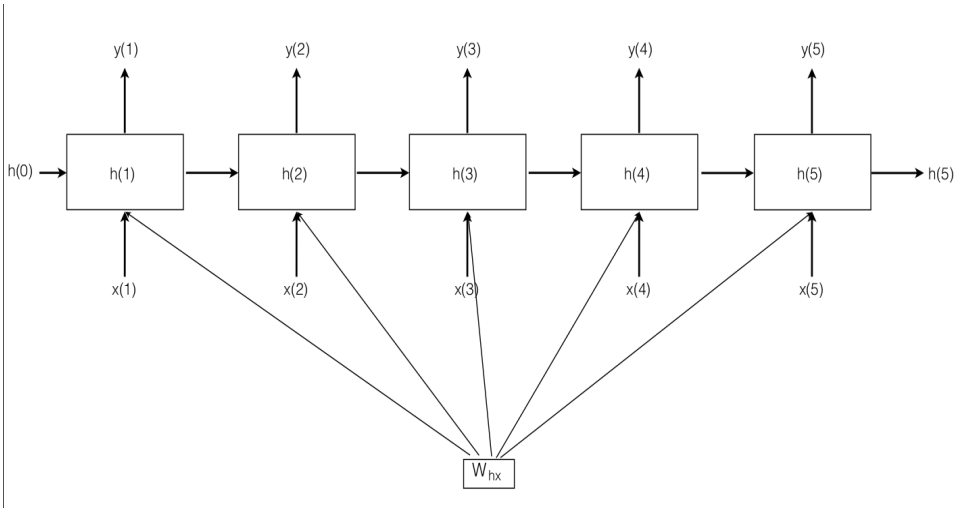

Figure IV.16: Representation of an unrolled Recurrent Neural Network.

Bidirectional RNNs

Until now, RNNs can be seen as a very powerful technique for prediction or transduction when the input is a sequence. Their strength comes from the fact that they model the previous elements in a sequence without making use of the Markov assumption, and so allowing the models to *remember* longer distance dependencies. A possible error is that some words may need the right context information to work better. Let's imagine that we have a RNN that translates a sentence from Spanish to English:

Un banco es una entidad financiera

If we check the word *banco* in a dictionary, we will find at least two possible translations, from which the most common would be "bank" or "bench". By taking a look at the previous words of *banco*, we cannot guess the correct translation of the word,[16] but if we take a look at the words in the right context, by seeing *entidad* and *financiera*, the system would have more information to disambiguate.

Bidirectional RNNs [Schuster and Paliwal, 1997] (BRNN) are models that exactly perform in that way, because they analyze an input sentence in both directions (forward and backward). That is the main basic idea, they

[16] "*un*" is the singular masculine determiner, equivalent to the determiner in English "a".

analyze a sequence from the left to the right and from the right to the left, and then the results are combined. This combination can be done in several ways, such as, linear pooling [Berger, 2013], logarithmic pooling [Jacobs, 1995] or just concatenation.

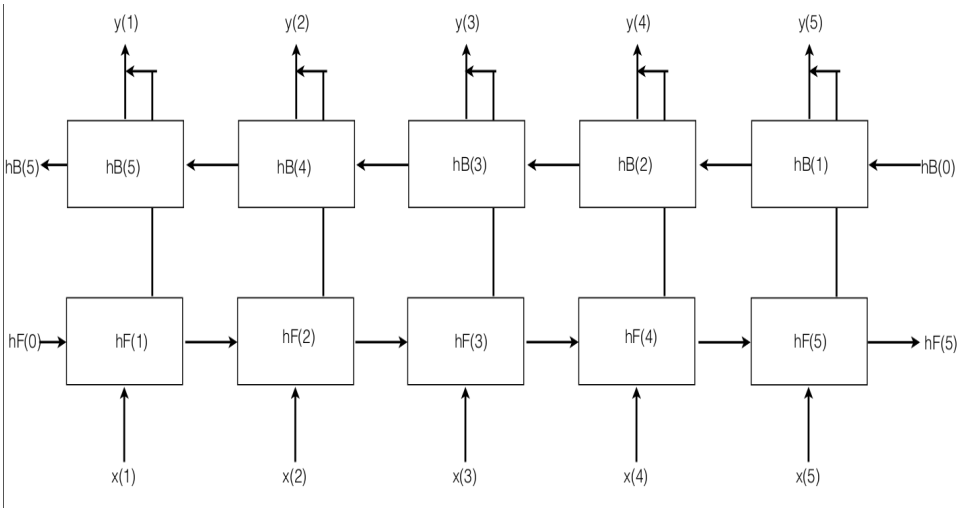

Figure IV.17: Representation of a Bidirectional Recurrent Neural Network that concatenates the output of the forward and backward RNNs.

As there are no interactions between the forward and backward RNNs, they can be unfolded and trained in the same way as regular recurrent nets.

LSTM for Deep Learning

As the inputs to a RNN are given and processed throughout time, the influence of such input in the hidden layers and the resulting outputs can either decay or blow up exponentially in long sequences. This problem is referred in the RNN literature as the vanishing gradient problem [Hochreiter et al., 2001]. Several solutions have been proposed to this issue and a popular solution is the architecture named Long-Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997].

The LSTM architecture gets an input and saves the current memory state just like in Recurrent Neural Networks. The difference between RNNs and RNNs with LSTM is that in the later ones input, output and forget gates (i_t, o_t and f_t in equation IV.26) are incorporated, in order to decide which

information should be remembered and which should not. The mathematical definition of an LSTM cell [Gers et al., 2000, Graves, 2013] is[17]

$$\begin{aligned} f_t &= \sigma_g(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ i_t &= \sigma_g(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ o_t &= \sigma_g(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_h(W_c x_t + U_c ht - 1 + b_c) \\ h_t &= o_t \circ \sigma_h(c_t) \end{aligned} \tag{IV.26}$$

where x_t is the input vector, h_t the output vector and c_t the cell state vector. The functions σ_h and σ_g represent nonlinear operations, specifically the hyperbolic tangent and the sigmoid. The operator $\circ$ is the element-wise product. U and W are the parameter matrices to be learnt.

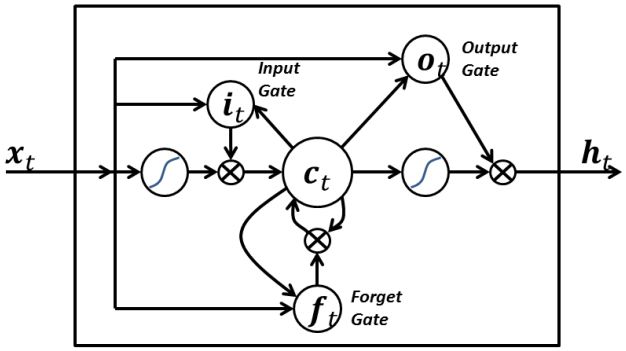

Figure IV.18: Long-Short Term Memory cell architecture.

IV.2.4.3 Embedding words or characters

As it has been seen, inputs to neural network models, either MLPs, RNNs or RNNs with LSTM, must be numerical. In Natural Language Processing, however, words or characters are a typical input, as we may want to work with a word or sentence. The bag-of-words model was proposed as a simple solution for this, where, given a vocabulary of M elements and an input sequence of N elements $T_1, T_2, \ldots T_N$, this is represented as a *bag* of M elements (vector of M elements), showing the frequency of each element in the input sequence.

[17] from Wikipedia, accessed on 28 Nov, 2016

A better solution is to use word embeddings [Mikolov et al., 2013c,b], where each word is represented in a dense D dimensional space. The most important characteristics of these vectors is that similar words have similar vectors. These representations are learned from untagged corpora and they are used for modeling semantic information of words. In order to learn them, current approaches rely on the distributional hypothesis [Harris, 1954]. Common software for training word embeddings include `word2vec`[18] [Mikolov et al., 2013a,b] and `GloVe` [Pennington et al., 2014].[19]

Deep Learning models that work with natural language commonly have a primary embedding layer in order to get the numeric representation of an input word. If pretrained word embeddings are incorporated, this embedding layer returns the previously calculated vectorial representation of the input word. When word embeddings are not available, these are randomly initialized and their parameters are learned jointly for the task.

In this work, when pretrained word embeddings were needed, I have used `word2vec` within the Gensim package[20] [Řehůřek and Sojka, 2010] to train them.

IV.2.4.4 Software for Deep Learning

In this subsection some specific models that follow the basic architecture introduced above are presented.

Frameworks

There are several frameworks that provide useful variables and functions for the development of Neural Network architectures. These are the main frameworks:

- DyNet [Neubig et al., 2017]
- Tensorflow [Abadi et al., 2016b,a]
- Theano [Theano Development Team, 2016]
- Blocks [Van Merriënboer et al., 2015]
- Caffe [Jia et al., 2014]

[18] `http://word2vec.googlecode.com`
[19] `http://nlp.stanford.edu/projects/glove`
[20] `https://radimrehurek.com/gensim/`

- Torch [Collobert et al., 2011]
- DeepLearning4J [DJD Team]

Recurrent Neural Network Language Models

`Char-rnn` is a software written in the Torch language that calculates recurrent neural network language models [Mikolov et al., 2010] which are character based. [Karpathy et al., 2015]

Sequence to sequence Encoder-Decoder models

Sutskever et al. [2014] proposed a Machine Translation model based on an Encoder-Decoder architecture and Bahdanau et al. [2014] an attention mechanism for it. This model has been implemented using several frameworks, such as Blocks[21] or Tensorflow.[22] This model was used in the SIGMORPHON 2016 shared task on Morphological Reinflection [Kann and Schütze, 2016] and it got impressive results [Cotterell et al., 2016]. This implementation was similar to Faruqui et al. [2015].

Bi-directional LSTM (Words+chars) + CRF layer

Lample et al. [2016] propose a complete independent architecture that extracts the information from the input graphemes and the joint information that those graphemes have when they compose a word. Furthermore, it has a CRF layer which takes into account the dependencies among outputs. This model is used for Named Entity Recognition without the use of external language-specific resources, such as gazetteers, POS-tags, ... and it reaches state-of-the-art performance in Named Entity Recognition in four languages.

Considering its architecture it fits perfectly for the task of poetry scansion, as it models each words character sequence, the interaction between words and also the conditional dependencies between output elements.

Broadly speaking, the model has two Bidirectional RNNs with LSTM and a CRF layer. The first Bidirectional RNN reads each input word in a forward and a backward fashion as it can be observed in table IV.5.

Each RNN that composes the BRNN is an encoder, which encode the words character-by-character information in the FWD_REPR and BWD_REPR variables, which later will be concatenated. So that to represent the whole

[21] Check the `machine_translation` example at `https://github.com/mila-udem/blocks-examples/`

[22] `https://www.tensorflow.org/versions/master/tutorials/seq2seq/index.html`

$$
\begin{array}{rl}
 & \begin{array}{ccccc} \mathrm{s} & \mathrm{w} & \mathrm{e} & \mathrm{l} & \mathrm{l} \end{array} \\
 & \mathrm{s} \rightarrow \mathrm{w} \rightarrow \mathrm{e} \rightarrow \mathrm{l} \rightarrow \mathrm{l} = FWD\_CHR\_REPR \\
BWD\_CHR\_REPR = & \mathrm{s} \leftarrow \mathrm{w} \leftarrow \mathrm{e} \leftarrow \mathrm{l} \leftarrow \mathrm{l}
\end{array}
$$

Table IV.5: Character modeling.

	to		*swell*		*the*		*gourd*		*and*		*plump*		*the*		*ha*		*zel*		*shells*
	↓		↓		↓		↓		↓		↓		↓		↓		↓		↓
FWD	to	→	swell	→	the	→	gourd	→	and	→	plump	→	the	→	ha	→	zel	→	shells
BWD	to	←	swell	←	the	←	gourd	←	and	←	plump	←	the	←	ha	←	zel	←	shells
	↓		↓		↓		↓		↓		↓		↓		↓		↓		↓
	O1		O2		O3		O4		O5		O6		O7		O8		O9		O10
	↓		↓		↓		↓		↓		↓		↓		↓		↓		↓
CRF	TAG1	↔	TAG2	↔	TAG3	↔	TAG4	↔	TAG5	↔	TAG6	↔	TAG7	↔	TAG8	↔	TAG9	↔	TAG10

Table IV.6: My caption

word, the model includes the previously mentioned word embeddings and it concatenates the input words embedding, together with the forward and backward representations (FWD_REPR and BWD_REPR). These word embeddings can be either pretrained (from an external corpus) or trained jointly for the current task.

Once that the words whole representation is built, there is another Bidirectional RNN that models the relationship among words. Whereas the previous Bidirectional RNN produced a single vector for each word (the output vectors size is constant, independent of the words length in characters), in this case, the BRNN produces an output for each word. This BRNN is similar to the one in figure IV.12. The outputs of each RNN that make up the BRNN (forward and backward) are concatenated.

Although the outputs $O1, O2, \ldots ON$, produced by these presented BRNNs, could be used directly, in this work these outputs go through a CRF layer, so that to model the dependencies among output tags. Ling et al. [2015] was a work in which the outputs were used without the CRF layer. They used these vectors as feature vectors for an independent classifier.

IV.3 Unsupervised learning

Two different paradigms for learning patterns have been seen so far. The first paradigm relies on manually written rules and the second on hand labeled data. Both of them are expensive as they require costly resources; either experts writing rules or human annotators.

Unsupervised learning is the task of finding the hidden underlying structure of unlabeled data. The main advantage of these models, in contrast with the supervised ones, is that there is no need of supervision or labeled data, and thus, big amounts of data can be cheaply treated. This structural information can be used in other processes. In recent years, Deep Learning models almost completely rely on unsupervised learning for the generation of word embeddings. In order to extract patterns from unlabeled data, some assumptions must be made. For example, as previously mentioned, the main assumption for learning word embeddings is the distributional hypothesis [Harris, 1954], which states that the meaning of a word is known by the company it keeps.

Currently unsupervised methods are also used to learn topic models. Topic models [Griffiths and Steyvers, 2004, Blei, 2012] are statistical models that model the underlying semantic structures that emerge in a set of documents. The assumption made by a topic model is that the words of a document are generated according to a probability distribution over the words and the main topic of each document is generated by another probability distribution over a number of possible topics. For instance, by selecting a topic about *astronomy*, the model will be more likely to generate words like *spacecraft*, *sky*, ... than others such as, e.g. *robbery*, *book*, ...

Almost all unsupervised learning can be understood as a clustering problem, where a set of input elements must be grouped in a number of clusters and the elements from the same cluster are supposed to be similar.[23] Typically used algorithms for unsupervised learning include K-Means and Expectation-Maximization. In this work, as the main interest is finding structure in sequential data, previously presented Hidden Markov Models are used to perform unsupervised analysis in sequences.

IV.3.1 K-Means algorithm

K-Means is a centroid-based clustering algorithm that groups a set of input instances into K clusters. This method is simple and effective. When it is started, K points are chosen at random as initial centroids/means for the clusters. Then, each instance is said to belong the nearest cluster.[24] After this assignment, the mean of each cluster is recalculated, as some instances may have changed. This procedure is repeated until stability is reached. We

[23] This similarity function has to be specified. For example in word embeddings, similar words are semantically similar.

[24] Several distance metrics could be used, such as, Euclidean distance, Manhattan distance, ...

reach stability when the same clusters are assigned to the same instances continuously. The pseudocode of the algorithm can be seen in listing IV.4.

Listing IV.4: K-Means algorithm.

```
CLUSTERS [1..K] = Randomly initialize centroids of K
    clusters
while not stable: //Iterate until the means are stable
  for each instance:
    //Set the nearest cluster to each instance
    setCluster (nearest(1..K), instance)
  for cluster in CLUSTERS:
    //Recalculate the mean of the cluster
    mean(cluster) = recalculateMean(instances(cluster))
```

Usages of the K-means algorithm in NLP.

IV.3.2 Expectation-Maximization

In the K-means algorithm, the clusters themselves are known beforehand (they are initialized randomly) and their specific location is changed according to the instances' means. The Expectation-Maximization algorithm —EM algorithm— [Dempster et al., 1977] works in a similar way as it also performs several iterations until convergence is reached. An important difference [Witten et al., 2016] from the K-means algorithm is that the parameter estimation is not made to the clusters themselves, but to the cluster probabilities (mean and standard deviation). For each instance x_i in the data, there is a weight value w_i that will express the probability of belonging to each cluster. Let's suppose that we have N data points and we need to divide the data into K clusters. The mean and standard deviation of the clusters would be calculated as:

$$\begin{aligned} Given: i \in (1..N); k \in (1..K) \\ w_{i,k} = P(x_i \in Cluster(k)) \\ \mu_k = \frac{w_{1,k}x_1 + w_{2,k}x_2 + \ldots + w_{N,k}x_N}{w_{1,k} + w_{2,k} + \ldots + w_{N,k}} \\ \sigma_k^2 = \frac{w_{1,k}(x_1 - \mu_k)^2 + w_{2,k}(x_2 - \mu_k)^2 + \ldots + w_{N,k}(x_N - \mu_k)^2}{w_{1,k} + w_{2,k} + \ldots + w_{N,k}} \end{aligned} \tag{IV.27}$$

In the EM algorithm the iterations finish when the overall likelihood of

the data is stable. This overall likelihood is calculated by multiplying the sums of the probabilities of the instances to belong to each cluster:

$$\prod_{i=1}^{N}\sum_{k=1}^{K} Pr(Cluster(k)) \times Pr(x_i|Cluster(k)) \tag{IV.28}$$

This function reflects the overall likelihood of the data to the proposed cluster probabilities. After performing several iterations, when the likelihood differences are sufficiently small, the process is said to be finished. As computing multiplications is computationally more expensive, log-likelihoods are employed (so that sums are used instead of multiplications). The EM algorithm is guaranteed to converge to a maximum, but this can be a local optimum.

IV.3.3 Hidden Markov Models

In the supervised learning section of this chapter, Hidden Markov Models were presented as simple generative models for performing sequence-to-sequence problems. The model itself is the same for supervised or unsupervised learning. The difference lies in the way parameters are learned. Let's recall that a Hidden Markov Model is a statistical model that has three elements:

1. Initial probabilities
2. Emission probabilities
3. Transition probabilities

In a supervised fashion, these parameters are learned using the maximum likelihood estimation (MLE). But as in unsupervised learning there is no tagged data, they are learned using the Baum-Welch algorithm. The Baum-Welch algorithm makes use of the previously mentioned EM algorithm [Dempster et al., 1977] and the forward-backward algorithm [Rabiner, 1989] and it is iterated until convergence is reached.

I have used two publicly available implementations of Hidden Markov Models, the one included in the NLTK package[25] and also treba[26] [Hulden, 2012].

[25] `https://github.com/nltk/nltk/`
[26] `http://treba.googlecode.com/`

IV.4 Discussion on methods

In this chapter several methods for tackling the problem of poetry scansion have been explored. The first one, implemented in the software called *ZeuScansion*, is a rule-based system and is only available for English. That is the usual shortcoming of rule-based systems, that rules must be rewritten for each language.

After that, I have reviewed a set of Machine-Learning based algorithms. Among these, supervised and unsupervised approaches were shown. Some of the supervised algorithms make independent predictions for each syllable (single/greedy prediction), and some others perform jointly and try to find an optimal solution (structured prediction) making them more suitable for sequence labeling problems. The disadvantage of supervised methods is that feature extraction must be done manually and trying different feature configurations can be costly. In this work, I will try the supervised methods for English language. Once the best performing methods are discovered for the English language, these will be extrapolated to other languages (Spanish and Basque).

Finally, some current Neural Network models have been explored. The main advantage of these frameworks is that, as it could be seen in this chapter, there are architectures that learn representative feature spaces directly from tagged data, so feature configurations do not need to be designed. The same methodology will be applied to these models; I will apply the best performing models to Spanish and Basque.

The ideal solution would be to use unsupervised approaches, as they do not need to have any tagged data nor rules, but usually they do not perform as well as supervised methods. Unsupervised models will be tested on English and Spanish data. Albeit there is interest on performing experiments on Basque data, I will not try unsupervised learning on Basque poems, as it is the first proposal of the annotated corpus.

CHAPTER V

Experiments and results

Once that we have seen the main information about the classifiers and methods that are employed in this work (chapter IV), it is time to check their performance on the corpora described in chapter III.

Let's recall that there are corpora in three different languages; English, Spanish and Basque. The corpus in English has been downloaded from an interactive website for scanning English verse [Tucker, 2011]. The Spanish one is from a corpus of the Spanish Golden Age [Navarro-Colorado et al., 2016] and the one in Basque is a subpart of a poetic anthology [Urquizu Sarasua, 2009], manually tagged.

The main idea here is to perform an intensive experimental study in the English corpus. The models that exhibit the best performance will then be applied to Spanish and Basque.

V.1 Experimental setup and evaluation

In order to evaluate rule-based systems, such as *ZeuScansion* or *Scandroid*, I tested them against the whole corpus, as they are expert systems and do not need a training dataset.

On the other hand, in the case of the supervised systems, I use a 10-fold cross-validation to train and test the models. In the Cross-Validation configuration, each of the 10 folds is divided in two parts: development part and testing part. I have used the development part to check the models validity and finally, the best models are tested on the testing part. Although creating a complete unseen testing dataset would be a more reliable solution

I had to rely on this evaluation method as the tagged dataset is not large enough.

When evaluating each of the annotated lines, the systems are evaluated by checking the error-rate obtained by using Levenshtein distance comparing each line from the automatically analyzed poem against each hand-made scansion from the Gold Standard [Graves, 2012, p. 13]. This is done in order not to penalize missing or superfluous syllables —which are sometimes present —with more than 1 count. For example, this line of poem by Henry W. Longfellow,

sent the wildgoose wawa northward

written in trochaic tetrameter, should be scanned as `/x/x/x/x`, while one of the tools —ZeuScansion—marks the line in question as `/x?/x/x`, as it can be seen in figure V.1.

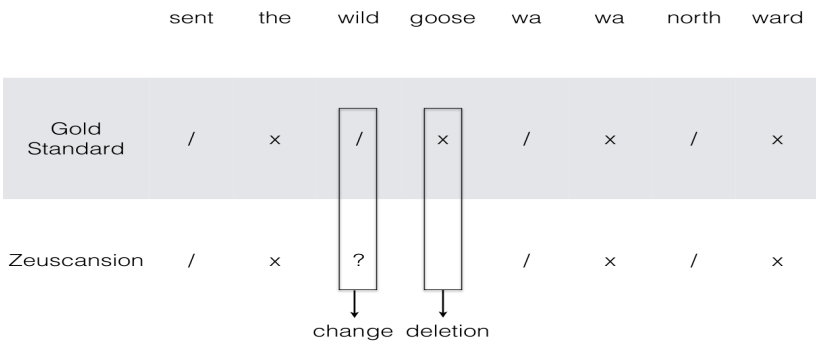

Figure V.1: In this figure a verse by Henry W. Longfellow can be seen. The second line shows how this line should be scanned and the third line an analysis proposed by *ZeuScansion*.

With the Levenshtein metric the distance between the analysis proposed by automatic tools, and the Gold Standard is calculated. Obviously, any proposed analysis identical to the Gold Standard will be assigned a distance of zero. The value that is obtained from using this distance metric can be interpreted as a minimum number of errors in the analysis.

In the previous example, *ZeuScansion* fails to assign the correct stress pattern to *wildgoose*, because the word does not appear in dictionaries and no similarly spelled word can be found. The minimum *Levenshtein* distance between the analysis and the reference is two, since changing the third `?` to a `/` and adding a `x` to the analysis would produce the 'correct' possibility in

the gold standard. Then, as the gold input has eight syllables, the evaluation system would return that the accuracy of the system is 0.75 $(1.0 - 2/8)$.

If more than one analysis are proposed in the Gold Standard, the distance to both of them is checked from the automatically predicted stress sequence and the minimum error is selected. For example in this excerpt from the sonnet "*Since there's no help*" by Michael Drayton [Palgrave, 1914]:

Be it not seen by either of our brows

one of the tools could scan it like

`/x//xxxxxx`

while in the gold standard the analysis of the line in question is

`xx//x/x/x/ | /x//x/xxx/`

The Levenshtein distance between these automatically and manually created results would be four and two, respectively. The evaluation metric would return that the proposed analysis has an error of 2 out of 10 syllables, which is the minimum error to get an acceptable scansion.

With this in mind, two accuracy metrics are used when evaluating the scansion models. The first one is the above presented per-syllable accuracy. The second metric, also used in works such as Gervas [2000], Greene et al. [2010], Navarro-Colorado [2017], checks the number of completely correct lines —i.e. when the Levenshtein distance between the gold standard and proposed analysis is zero.

V.2 Baselines

A baseline is a simple way to establish a lower bound for a statistical learning systems performance. Algorithms that are used as baselines are usually the most naive or trivial approaches to a problem, and in this case, I have defined seven heuristics[1], which are listed below:

- Always stressed: This approach will assign a stressed value to all instances

[1]Their code is available under the GNU/GPL license at `https://bitbucket.org/manexagirrezabal/baselines4scansion`

- Always unstressed: This approach will assign an unstressed value to all instances

- Lexical stress: In this approach stress is assigned to each syllable by checking its lexical stress. The issue here is that a three-level marking is used for lexical stress and the resulting stress should be in a two-level marking. If the lexical stress is secondary, they will be tagged as unstressed (the accuracy score was slightly higher). In the case that a word has X syllables and the lexical stress has less syllables —this can happen in the case of OOV words— the baseline adds a set of special characters at the end. E.g. if the word different is assigned the lexical stress "`/x`" and the syllable division states that the word has three syllables (dif-fer-ent), the baseline would assign the stresses as "`/xl`".

- Syllable weight: This approach will assign stress to heavy syllables and other syllables will be unstressed. It uses a naive algorithm for calculating the weight of the syllable[2].

- iambic/trochaic (based on the last syllable): A sequence of constant stressed/unstressed classes is assigned taking into account the last syllables lexical stress. If the last syllables lexical stress is stressed, then stress will be assigned as a sequence of continuously changing stress (stressed-unstressed-stressed-unstressed...).

- iambic/trochaic (based on the first syllable): The same is done, as in the previous case, but taking into account the first lexical stress to start assigning.

- Naive Bayes: This approach estimates the probability of having a stressed or unstressed class, given a syllable. This baseline expects a syllabified sequence of words.

Each of these baselines is applied to the English dataset as a whole. But, the Naive Bayes baseline is evaluated using a 10-Fold Cross-Validation, as it needs to learn the conditional probabilities of each stress given a syllable. In table V.1 you can see the output of each baseline given a poetic line.

In table V.2 the accuracies of these baselines can be seen. From these baselines some conclusions can be drawn. As the corpus is mainly iambic, just by assigning stress to the syllables following a constant iambic/trochaic

[2]If a syllable ends in consonant or it has a diphthong (simple list, without performing g2p). Algorithm in appendix B.

	wo	*man*	*much*	*missed*	*how*	*you*	*call*	*to*	*me*	*call*	*to*	*me*
Always stressed	/	/	/	/	/	/	/	/	/	/	/	/
Always unstressed	x	x	x	x	x	x	x	x	x	x	x	x
Lexical stress	/	x	/	x	/	/	/	x	/	/	x	/
Syllable weight	/	/	/	/	x	/	/	x	x	/	x	x
Iambic/trochaic (last)	x	/	x	/	x	/	x	/	x	/	x	/
Iambic/trochaic (first)	/	x	/	x	/	x	/	x	/	x	/	x
Naive Bayes	/	/	x	/	/	x	/	x	x	/	x	x
Gold Standard	/	x	x	/	x	x	/	x	x	/	x	x

Table V.1: Output of each baseline for an excerpt from "*The voice*" by Thomas Hardy.

Baselines	Per syllable (%)	Per line (%)
Always stressed	50.88	0.27
Always unstressed	49.76	0.00
Lexical stress	73.57	5.76
Syllable weight	62.14	0.55
Iambic (Based on first)	75.89	11.25
Iambic (Based on last)	83.64	**33.30**
Naive Bayes	**85.31**	26.93

Table V.2: Accuracies of baselines in the English dataset.

pattern, a quite high accuracy can be reached. There is an important difference between accuracies when the assignment process is started from the beginning of the line and the end of the line (5th and 6th rows). This benefit is got especially because of the effect of rhyme in poetry. Rhymed lines in poetry must follow the same rhythmic pattern, and thus, metrical variability in the last syllables' stress structure drops, compared to other syllables in a line.

Also, another deduction that can be made here is that considering the surrounding predictions of the current instance gives a higher accuracy, because, for example, the baseline that gives the highest accuracy is the one that assigns constantly changing stresses.

Program	Per syllable (%)	Per line (%)
ZeuScansion	86.17	29.37
Scandroid	**87.42**	**34.49**

Table V.3: Accuracies of rule-based systems

V.3 Rule-based scansion

As mentioned in the previous chapter, a rule-based system —*ZeuScansion*— was developed to perform scansion. This tool was presented in Agirrezabal et al. [2013c, 2016b] and released under the GNU GPL license on GitHub.[3]

In table V.3 the per syllable and per line accuracies of *ZeuScansion* can be seen.[4] The per syllable results are quite promising as it achieves a pretty good result by just using simple rules and no sequence information at all. Taking a look at the per line accuracies, it can be said that almost all lines have a small number of errors, which leads to lower this accuracy while the per syllable accuracy is sufficient. Getting sequence information such as the previous syllables or predictions, for example, could help to reduce this error. Although they could be better, I think that these results are promising as neither sequential (surrounding stresses) nor metrical information (whether the line is iambic, dactylic, ...) is used. The only pieces of information are words, their lexical stresses and their POS-tags.

The global analysis system —which calculates a poems meter according to the predominant feet— was also evaluated using two different works of poetry. The first one is Longfellow's *The song of Hiawatha* and the second one Shakespeare's *Sonnets* to check if ZeuScansion was able to infer the meter correctly. The meter of each sonnet in Shakespeare's writing (154 sonnets) was calculated; in the case of Longfellow's poem each stanza (637 stanzas) was analyzed separately. Shakespeare's sonnets are written in iambic pentameter and *The song of Hiawatha* in trochaic tetrameter. Table V.4 reports the accuracy on this task.

[3] `https://github.com/manexagirrezabal/zeuscansion`

[4] These results are slightly different from Agirrezabal et al. [2013c, 2016b] in that the Gold Standards version used for evaluation is newer. In Agirrezabal et al. [2013c, 2016b] a corpus downloaded in 2013 from the 4B4V website was used.

[5] 44.58% were classified as amphibraic dimeter.

Poem	Correctly classified (%)
The song of Hiawatha	32.03[5]
Shakespeare's Sonnets	70.13

Table V.4: Evaluation of the global analysis system (only ZeuScansion)

V.4 Supervised learning

Let's now turn into the supervised learning systems section. As it was presented in chapter IV, the employed models include greedy classifiers, structured predictors and Neural Networks.

V.4.1 Single/greedy prediction

Greedy classifiers, the ones that do not optimize the output predictions, were used. The employed classifiers were:

1. Naive Bayes
2. Linear SVM
3. Perceptron

One experiment that only included basic (and possibly the language-agnostic) features was performed, and subsequently another one, which included all the features presented in the previous chapter (subsection IV.2.1). Recalling the above, the basic feature configuration was composed of the first ten feature templates that included:

- Syllable position in line
- Syllable position in word
- Number of syllables in the line
- Phonological weight of syllable
- Words length
- Words last character
- Words last two characters
- Words last three characters

- Words last four characters
- Words last five characters

Training greedy sequence predictors with these attributes shows the basic capability of these predictors using little or (almost) no linguistic information. All these results are compared with the rule-based system ZeuScansion [Agirrezabal et al., 2013c, 2016b]. Results with this feature configuration can be seen in Table V.5 and it seems that quite acceptable accuracies can be reached by simply extracting basic attributes from words.

	Per syllable (%)	Per line (%)
ZeuScansion	**86.17**	29.37
Naive Bayes	78.06	9.53
Linear SVM	83.50	22.31
Perceptron	85.04	**28.79**

Table V.5: Accuracies of different classifiers using just the basic features (10 features) previously presented using 10-fold Cross-Validation.

As also stated previously, additional feature templates are used together with surrounding elements to model the current syllables context in a line of verse. These elements are included among the additional feature templates:

- Current syllable ($syllable[t]$) and surrounding $t \pm 10$ syllables
- Current word ($word[t]$) and surrounding $t \pm 5$ words
- Current POS-tag ($POS - tag[t]$) and surrounding $t \pm 5$ POS-tags
- Current lexical stress ($LS[t]$) and surrounding $t \pm 5$ lexical stresses

The results of the classifiers using all the features are reported in Table V.6. Here, both the SVM and the Perceptron see their scores improve significantly. In the case of the Naive Bayes classifier results do not improve as much as in the other cases, probably because of the sensitivity to overlapping features in Naive Bayes. The difference between the linear SVM and the Perceptron, especially in per-line accuracy, is somewhat noteworthy. Normally, the SVM, which finds a maximum-margin classification boundary, would be expected to outperform the averaged Perceptron, but that is not the case here in both the basic feature set experiment and the full feature set one.

	Per syllable (%)	Per line (%)
ZeuScansion	86.17	29.37
Naive Bayes	80.96	13.51
Linear SVM	87.42	34.45
Perceptron	**89.12**	**40.86**

Table V.6: Accuracies of different classifiers using all the features (64 features) presented above using 10-fold Cross-Validation.

V.4.2 Structured prediction

As single predictors do not optimize the resulting sequence labeling, they can make simple errors that propagate throughout the line—something that could be avoided by looking at the surrounding outputs. This is the main weakness of not using structured prediction systems. Two common structured prediction models were used:

1. Hidden Markov Models (HMM)
2. Conditional Random Fields (CRF)

In table V.7 the per line and per syllable accuracy of structured prediction systems can be observed (HMM and linear-chain CRF) compared to two rule-based systems. In the experiments, although the per-syllable accuracies do not vary too much, the per-line scores improve substantially by the use of structured predictors. The HMM has been trained in the standard way, that is, using single syllables (emissions) and their corresponding classes (states). The first CRF model is trained analogously, i.e. using only syllables as the features, and the previous label.

In the same table, the results of the CRF models using the aforementioned feature configurations (basic feature templates and additional feature templates) are reported. As expected, training the CRFs using the richer feature configurations employed in the greedy sequence predictors above yields a much higher accuracy, especially in the per line measure.[6]

V.4.3 Neural Networks / Deep Learning

In the previous chapter, in subsection IV.2.4, a review about neural network literature was done. Several models and architectures were presented and

[6]The results shown in this subsection and the previous one are published in Agirrezabal et al. [2016a]

	Structured prediction models		
	#FTs	Per syllable (%)	Per line (%)
ZeuScansion	-	86.17	29.37
Scandroid	-	87.42	34.49
HMM_{just_syll}	-	90.39	48.51
CRF_{just_syll}	1	88.01	43.85
CRF_{basic}	10	89.32	47.28
$CRF_{additional}$	64	**90.94**	**51.22**

Table V.7: Accuracies of structured prediction models using different sets of features on 10-Fold Cross-Validation. The second column expresses the number of feature templates used in the model.

from them, in this work I carried out experiments with the following systems:

1. Multilayer Perceptron [Witten et al., 2016]
2. Recurrent Neural Network Language Models
3. Encoder-Decoder architecture
4. Bidirectional Recurrent Neural Networks + CRF

Although Feedforward networks (Multilayer Perceptron) perform well in some cases, their performance is usually similar to classifiers like SVM or Perceptron, as they are not structured predictors. The obtained results using these models were lower than previous greedy predictors, so they were left aside. Also, the memory requirement of a Multilayer Perceptron is very high.

Some initial experiments were performed using Recurrent Neural Network Language Models (RNNLM), just by inserting a tag after each word (plus a special character). This tag was the stress that the word would have. A language model was trained in that way and it tried to predict the unseen tags. Resulting accuracies were below 80%, which in some cases was because the first prediction was not done correctly, namely because of error-propagation. Then, this model was not promising for the task.

Encoder-Decoder

As an Encoder-Decoder architecture gave high quality results in several tasks, such as Machine Translation [Sutskever et al., 2014, Bahdanau et al.,

	Per Syllable (%)	Per Line (%)
S2S	84.52	30.93
W2S	**85.44**	**34.00**

Table V.8: Best results of the Encoder-Decoder model in the English dataset (development set).

2014] or morphological reinflection [Kann and Schütze, 2016], experiments were performed with such architecture.

The problem was modeled as a character sequence to stress sequence problem. As explained earlier in chapter IV, in the Encoder-Decoder architecture the input sequence is compressed to a fixed-size vector and then, the decoder is the responsible of producing the output. Because of that, as there is no restriction about the number of input and output elements, two specific experiments were accomplished.

The first experiment was just to learn a mapping from a space-separated syllable sequence to a stress sequence (**S2S**),[7] e.g.:

the in vis i ble worm → `xx/xx/`

But as word structure is important, the second experiment was to learn from a syllabified word sequence to a stress sequence (**W2S**),[8] for instance:

the in.vis.i.ble worm → `xx/xx/`

In these experiments there was no improvement over the results of previous models. Despite testing the model with different hidden layer sizes and embedding sizes, results were not sufficient. In table V.8 the best obtained results can be seen. These optimal results were got using 20 hidden units in both the encoder and the decoder. In the case of the character and the output tag embedding size, they were set to 40.

Bidirectional LSTM + CRF

As reviewed in the subsection IV.2.4 from previous chapter, the model presented in Lample et al. [2016] models the character sequences in each word and models those word representations throughout a sentence. Furthermore, semantic representations can be included in the form of word embeddings,

[7] Syllable-to-stress
[8] Word-to-stress

to check whether semantic/syntactic information extracted from corpora (without supervision) is helpful in the metrical analysis of poetry. Because of that, the problem was represented using two different methods (as in subsection II.3.3):

1. Find a mapping from words to stress patterns (**W2SP**)
2. Find a mapping from syllables to stresses (**S2S**)

The first way tried to find a mapping from words to stress patterns (**W2SP**). The second method represented verse lines divided by syllables and for each syllable a binary task should be performed, marking each syllable as stressed or unstressed, syllable to stress (**S2S**). In both cases, a Bidirectional LSTM reads the character sequence from each syllable or word and together with its own embedding, a joint representation is built. Then, there is a word-level Bidirectional LSTM that reads each word or syllable, and produces an output.[9] The main reason for performing the first experiment is that pretrained word embeddings can be incorporated.

Verse	Stress sequence
Therefore my verse to constancy confined,	[+-] [-] [+] [-] [+-+] [-+]
There fore my verse to cons tan cy con fined,	[+] [-] [-] [+] [-] [+] [-] [+] [-] [+]

The results of these two methods can be seen in table V.9. By looking at the results, it was quite evident that giving the verse lines syllabified assisted the model to extract the phonological structures of each syllable and thus, results were better. From these initial results two different paths were explored:

1. Incorporate word embeddings (in the W2SP model)
2. Add the Word Boundary to the syllables (in the S2S model)

	Per Syllable (%)	Per Line (%)
W2SP	90.80	53.29
S2S	**93.06**	**61.95**

Table V.9: Initial results using the Bi-LSTM-CRF.

[9] If the problem is modeled as a S2S problem, the output dictionaries size will be 2 (stressed or unstressed).

	Per Syllable (%)	Per Line (%)
W2SP-NoPretr	90.80	53.29
W2SP-PretrLiter-W=2	91.46	52.49
W2SP-PretrLiter-W=5	90.54	49.91
W2SP-PretrWiki-W=2	91.06	51.48
W2SP-PretrWiki-W=5	91.42	53.57

Table V.10: Results of the Bi-LSTM-CRF including pretrained word embeddings.

In order to incorporate pre-trained vector representations of words (word embeddings), these were calculated from a larger corpus. Two corpora were used, English Wikipedia (**PretrWiki**) and some literary works (**PretrLiter**) downloaded from Project Gutenberg.[10] The embeddings were trained using 300 dimensions for the word vectors and leaving all the other parameters as default.

As including syntactic information is useful in order to calculate the stress values word embeddings were expected to assist in this. Then, two different window sizes were used to train the word representations, **W=5** and **W=2**.[11] The results of these experiments can be seen in table V.10 and as can be seen, results are not conclusive.

As stated before, the second mentioned path was to add word boundary information. Sometimes, having all the words divided into syllables, makes it difficult to calculate the possible stress sequence, as the sequence of words becomes obscure, or hard to infer. Let us consider two verses from Shakespeare's *Sonnets*, No. 6 and 60, respectively:

Verse	Syllabified verse
that's for thyself to breed another thee	*that's for thy self to breed a no ther thee*
in sequent toil all forwards do contend	*in se quent toil all for wards do con tend*

In the second example, the syllable "*for*" is known to carry stress because is part of the word "*forwards*". However, if it is an independent word, it could be both stressed or unstressed. Because of this, a word boundary marker (**Word_Bound**) was included in each syllable if it was the last syllable of the current word. This improved the results, as can be seen in table V.11.

[10]The downloaded literary works were from the same authors included in the 4B4V corpus.

[11]Smaller window sizes are supposed to model syntactic information of a word. Bigger window sizes capture broad topical content about a target word [Goldberg, 2015].

	Per Syllable (%)	Per Line (%)
S2S	93.06	61.95
S2S + Word_Bound	**94.49**	**69.97**

Table V.11: Results of the Bi-LSTM-CRF including word boundaries.

	#FTs	Per Syllable (%)	Per Line (%)
ZeuScansion*	-	86.17	29.37
Scandroid* [Hartman, 2005]	-	87.42	34.49
Iambic (Based on last)*	-	83.64	33.30
Naive Bayes*	-	85.31	26.93
Perceptron_{10} (S2S)	10	85.04	28.79
Perceptron_{64} (S2S)	64	89.12	40.86
HMM (S2S)	-	90.39	48.51
CRF_{10} (S2S)	10	89.32	47.28
CRF_{64} (S2S)	64	90.94	51.22
Bi-LSTM+CRF (W2SP)	-	89.39	44.29
Bi-LSTM+CRF (S2S)	-	91.26	55.28
Bi-LSTM+CRF+WB (S2S)	-	**92.96**	**61.39**

Table V.12: Results of the best classifiers in the English dataset (testing set) compared to rule-based approaches and baselines.

After doing all the above experiments in the development data, results of the most promising models on the testing data are shown, so that to see the validity of the parameters previously analyzed and chosen. In table V.12 all these results can be seen.

To conclude with experiments in English language a last model was trained. Previous methods using syllabic information had a limitation in their model, latent also in some of the feature-based predictors. Syllabification was assumed to be done beforehand. As we were aware of such limitation which was not present in ZeuScansion, a last system was created. This system performed several subtasks to scan poetic lines completely automatically from raw texts. The aim of this experiment was to show that the Bidirectional LSTM model could be also used with automatically syllabified poetry. In the following lines the steps that this model performs can be seen:

1. Grapheme2Phoneme conversion using the software package *Phonetisaurus* [Novak et al., 2012] trained on the NETTalk pronunciation

	Per Syllable (%)	Per Line (%)
ZeuScansion	86.17	29.37
W2SP	90.80	53.29
W2Phon2Syll2Str	**94.79**	**71.04**

Table V.13: Results of the model that performs grapheme-to-phoneme, syllabification and Neural Network based scansion in the English dataset (development set), compared to previous models.

dictionary [Sejnowski and Rosenberg, 1987].

2. Syllabification and word boundary marking [Hulden, 2006] using the software *Foma* [Hulden, 2009].

3. Bidirectional LSTM + CRF based scansion (syllable-to-stress, S2S) [Lample et al., 2016].

The results of this model in the development data compared to previous W2SP models are shown in table V.13. Even though there is a previous preprocessing part, results are still sufficiently good. In fact, the best results in the development data were got in this case.

In order to see that the BiLSTM+CRF model learns phonological patterns from poetic data, in figure V.2 the output of the Bidirectional LSTMs can be seen, showing the first line of the poem "*Scrambled Eggs Super!*", by Dr. Seuss. The columns that represent the stressed syllables (2nd, 5th, 8th and 11th syllables) stand out clearly. This representation is used as input for the CRF layer, which finds the optimal resulting sequence.

V.5 Extrapolating to Spanish

After doing all the experiments in English, the previous data-driven techniques were applied in the Spanish dataset, so as to see the applicability of the best models and their features. Firstly, the accuracies of some baselines were calculated, specifically, the best four baselines. Table V.14 shows these results.

Results of this table illustrate the high bias toward the lexical stress, which can be because it is used in the heuristic to equal the number of syllables and stresses (see section III.3 and appendix A). Probably because of that, a high accuracy is reached with the lexical stress baseline.

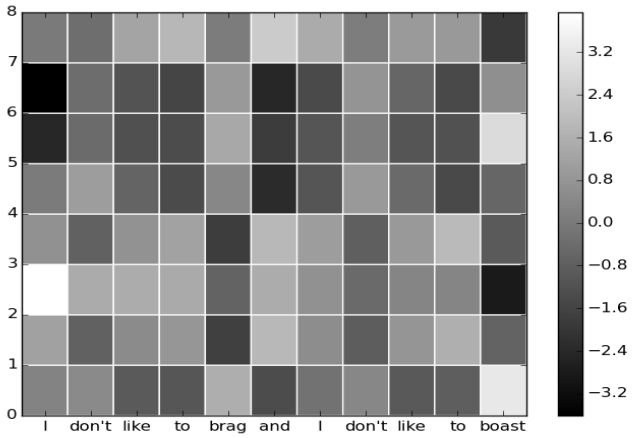

Figure V.2: Output of the Bidirectional LSTM (8 different outputs) trained on English poetry. These values are the input of a CRF layer. The input sentence is "*I don't like to brag and I don't like to boast*".

Baselines	Per syllable (%)	Per line (%)
Lexical stress	**94.11**	**50.00**
Iambic (Based on first)	73.78	0.79
Iambic (Based on last)	75.22	0.84
Naive Bayes	82.20	6.37

Table V.14: Accuracies of baselines in the Spanish corpus.

When using the Naive Bayes baseline, the prior probability of having a stressed syllable was checked, and that was different from the English corpus. The prior probability of a syllable to be unstressed was 30 points higher than being a stressed one. Almost 66% of the syllables in the Spanish corpus are unstressed.

When performing experiments with the supervised classifiers, some models were left aside because of their low results, such as, Naive Bayes and Support Vector Machines with 10 and 64 features.

Results in table V.15 show that previous work [Gervas, 2000, Navarro-Colorado, 2017] is hard to beat and also that neural methods outperform feature-based classifiers and structured predictors (similar as in English).

It can be observed in table V.15 that the 10 basic and hypothetical

	#Features	Per Syllable (%)	Per Line (%)
Navarro-Colorado [2017]	-	-	**94.87**
Gervas [2000]*	-	-	**88.73**
Lexical stress	-	**94.11**	**50.00**
Naive Bayes	-	82.20	6.37
Perceptron	10	74.39	0.44
Perceptron	64	91.49	35.71
HMM	1	92.32	45.08
CRF	10	84.89	18.61
CRF	64	92.877	55.44
Bi-LSTM+CRF (W2SP)	-	**98.95**	**90.84**
Bi-LSTM+CRF (S2S)	-	95.13	63.68
Bi-LSTM+CRF+WB (S2S)	-	98.74	88.82

Table V.15: Results of the best classifiers in the Spanish dataset (testing set) compared to rule-based approaches and baselines.

language agnostic features do not work very well when used with a greedy model, such as the Perceptron. But, if they are used in a CRF model, as they use the output sequence information too, it seems that results get improved slightly, but not enough.

The inclusion of additional features, especially the current word information gives a good cue to calculate the stress of the current syllable. In order to see that word information is crucial in Spanish scansion, please refer to Bi-LSTM + CRF results, one of them including word boundaries **+WB** and the other not including them. The per-syllable accuracy has an increase of approximately 3 points and the per-line accuracy goes from 63.68 until 88.82.

To conclude with the extrapolation of experiments, the encoder-decoder model was used in the Spanish dataset. The main motivation for using such model was its flexibility in the input sequence and output sequence lengths (Both sequences length does not need to be the same). Hence, the encoder-decoder would learn when to perform synaloephas without making use of any heuristic. A per-syllable and per-line accuracy of 89.89% and 47.32% was reached, respectively.

Baselines	Per syllable (%)	Per line (%)
Lexical stress	25.00	0.15
Iambic (Based on first)	**81.15**	**29.74**
Iambic (Based on last)	75.86	12.88
Naive Bayes	66.15	0.64

Table V.16: Accuracies of baselines in the Basque corpus

V.6 Extrapolating to Basque

As you recall from chapter III, the Basque corpus was composed of two sub-corpora; one with annotated poems according to a sung version, and the other according to how poems are recited.

Preliminary experiments were conducted using the whole corpus and after that with each subpart. Experiments with the whole corpus showed a high variability between folds, and thus, reliable conclusions could not be made. The same happened with the chanted sub-corpus. Then, because of the stability of results, the recited portion of the whole annotated corpus (38 poems) has been used.

The accuracies of the same baselines as in Spanish were calculated and these results are reported in table V.16. The iambic baseline gives the best results, based on the first lexical stress. It is noteworthy to check the low result that the lexical stress baseline shows.

This could be understood because in standard Basque stress is assigned to the [+2,-1] syllables[12] according to Hualde [1994]. But when stress is assigned to a sentence or whole utterance this [+2,-1] is extended to the sentence or prosodic group, so, for instance, in the word "txistularia" prominences should be assigned as "txis-tú-la-ri-à",[13] but the sentence "txistularia da" should be realized as "txis-tú-la-ri-a dà" (from [Hualde, 1994, p. 1559]).

Following the steps of the Spanish extrapolation, the same experiments were replicated in the Basque poetry dataset. Results are reported in table V.17. These results are more homogeneous than previous results in other languages. There is a notable improvement with the usage of structured prediction methods, as it can be appreciated in the results. But all the other results show a slight variation, which are not significant according

[12]Primary stress is assigned to the second (+2) syllable and secondary stress is assigned to the last syllable (-1).

[13]In this case, stresses are marked with diacritic symbols. Primary stress with acute accent ($\acute{a}$) and secondary stress with grave accent ($\grave{a}$).

	#Features	Per Syllable (%)	Per Line (%)
Iambic (Based on first)	-	81.15	**29.74**
Iambic (Based on last)	-	75.86	12.88
Perceptron	10	71.77	9.74
Perceptron	64	69.86	8.47
HMM	1	80.97	24.10
CRF	10	81.19	26.23
CRF	64	80.52	26.93
Bi-LSTM+CRF (W2SP)	-	**83.19**	23.75
Bi-LSTM+CRF (S2S)	-	79.38	20.32
Bi-LSTM+CRF+WB (S2S)	-	79.66	24.67

Table V.17: Results of the best classifiers in the Basque dataset (recited) compared to baselines.

to a Welch's t-test. This low significance is distinguishable because of the high variance that the cross-validation results manifest. In the table, it is remarkable that the per line accuracy of the iambic baseline was not beat.

V.7 Unsupervised learning

Toward the completely unsupervised analysis of poetry, a simple cross-lingual experiment was tried. Using the Bidirectional LSTM framework, a model was trained on Spanish data and was tested against the English corpus. The same was done in the opposite direction. The intuition for this attempt was that some typical structures among characters would be shared across languages and the Deep Learning model would be able to learn them. The per-syllable results were between 68.95% and 71.65% for each language pair. The per-line accuracies were too low (below 1.7%).

Apart from this, typical clustering algorithms presented before— K-Means and EM— were tested in the English dataset. We took the dataset with 64 feature templates, applied the usual filters in order to convert strings to numeric values and evaluated both algorithms. The per-syllable accuracies were below 55%.

Finally, Hidden Markov Models were used for unsupervised scansion. In order to check the models learning capacity, first order models with two different states (unstressed/stressed) were trained. The obtained output was not representative and thus results were too low. The intuition was that using

higher order models would help,[14] and to that end models with more states (4, 8 and 16) were trained, simulating higher order models. As unsupervised learning models results can vary among experiments, each experiment was repeated 100 times for each number of states. Also, it is worth to mention that each model, getting a syllabified poem line as input, returns a numeric sequence. The numbers from that sequence range from 0 to $N-1$, being N the order of the HMM.

In tables V.18 and V.19, the average accuracy of each model is reported for each language compared with the best baselines, rule-based models and the best previously mentioned guesser. Results clearly show that accuracies get improved when higher order models are used (until 16 states). Sometimes this improvement is important and in other cases slight. But the most interesting fact is that the standard deviation —i.e. variability— of the results is reduced with higher ordered models.

	Per syllable (%)	Per line (%)
Iambic (Based on last)	83.64	33.30
Naive Bayes	85.31	26.93
ZeuScansion	86.17	29.37
Bi-LSTM+CRF+WB (S2S)	92.96	61.39
HMM (4 states)	66.28	7.29
HMM (8 states)	74.65	9.91
HMM (16 states)	**76.51**	**12.53**
HMM (32 states)	74.03	8.07

Table V.18: Accuracy of unsupervised Hidden Markov Models in the English dataset, compared to previous models.

[14] It is evident that to model dactylic (`/xx`) or anapestic (`xx/`) poems at least two previous stresses have to be checked.

	Per syllable (%)	Per line (%)
Lexical stress	**94.11**	**50.00**
Naive Bayes	82.20	6.37
Navarro-Colorado [2017]	–	94.87
Gervas [2000]	–	88.73
Bi-LSTM+CRF (W2SP)	98.95	90.84
HMM (4 states)	68.60	0.13
HMM (8 states)	76.40	2.93
HMM (16 states)	**76.77**	**3.05**
HMM (32 states)	75.62	2.34

Table V.19: Accuracy of unsupervised Hidden Markov Models in the Spanish dataset, compared to previous models.

CHAPTER VI

Discussion and Future Directions

In this work several methods for automatic poetic scansion have been presented. These models can be rule-based or data-driven. The main advantage of data-driven systems is their high applicability to other languages, if tagged data are available. In this work, the main investigation has been carried out in English, and then, the best resulting models have been applied to Spanish and Basque.

As a rule-based system, we presented ZeuScansion, a tool for scansion of English poetry [Agirrezabal et al., 2016b]. Subsequently, experiments have been carried out in English by using supervised methods. Among these supervised methods, three types of learners have been considered: independent predictors, structured predictors and Deep Learning models.

After training and testing models for poetic scansion in English, models for Spanish and Basque have been generated. Together with these language-specific methods, unsupervised learners were tested, which learn patterns from untagged (raw) data, only in English and Spanish data.

This section includes a review of results and corresponding conclusions. Also included, is a discussion on research questions posed in the introduction of this thesis. The section closes with a summary of the contributions of this work and suggestions for further research.

VI.1 Conclusions

The experimentation carried out in this work gives an insight into poetic traditions and some conclusions can be made. Such conclusions are presented below.

The baselines reveal some characteristics of the studied languages' poetic traditions. Some poetic traditions rely strongly on the lexical stress (Spanish poetry); this is evidently observed in the high accuracy of the lexical stress baseline. On the other hand, in English poetry, certain syllables are always stressed or unstressed, unlike in Spanish. Evidence for this is the high prediction accuracy of the Naive Bayes baseline.

Later, the first presented method for automatic scansion is *ZeuScansion* [Agirrezabal et al., 2013c, 2016b]. ZeuScansion correctly predicts the stresses in the 86.17% of the syllables and predicts correctly 29.37% entire lines. Considering that contextual elements are not used for prediction, and that a stress pattern is therefore produced for each word independently, results are promising. This result, however, is lower than the other rule-based system presented in Hartman [2005], *Scandroid*, which achieves an 87.42% per-syllable accuracy and a 34.49% per-line accuracy.

Moving toward data-driven methods, after performing a diverse number of experiments, previous rule-based methods have been improved upon. The inclusion of structural information, i.e. the use of structured prediction models, significantly improves preceding results.

Supervised learning models have resulted in high accuracy, reaching a minimum of 80% of accuracy for all languages. In the English data, per-syllable results are similar to those reported in Estes and Hench [2016] (F1-Score of 0.904 on held-out testing data) for the supervised analysis of Middle High German poetry.

Among data-driven systems, it is worth mentioning the high results that the Bidirectional LSTM yield, without making use of any hand-crafted features. The Neural Network learns useful patterns directly from data, by learning information from character sequences along with word and phrase information, which suggests that they model the phonological structure of words within a context.

Spanish results are not so competitive. In Gervas [2000], a per-line accuracy of 88% was achieved. However, recently published results [Navarro-Colorado, 2017] show a per-line accuracy of 94.87%, while the best per-line accuracy achieved in this work is 90.84% for Spanish. One of the shortcomings of the best scansion model presented here (Bi-LSTM+CRF) is that it requires one output for each input, which makes it challenging to create systems in languages where synaloephas are used, such as in Spanish.

Results obtained in the Basque dataset should be considered preliminary. Because of the high variability, reflected in a rather high standard deviation, not too many conclusions can be made. The use of structured prediction methods gives the best per-syllable results in Basque data. It is noteworthy

	English	Spanish	Basque
Perceptron10	85.04	74.39	71.77
Perceptron64	89.12	91.49	69.86
HMM	90.39	92.32	80.97
CRF10	89.32	84.89	81.19
CRF64	90.94	92.87	80.52
Bi-LSTM+CRF (W2SP)	89.39	**98.95**	**83.19**
Bi-LSTM+CRF (S2S)	91.26	95.13	79.38
Bi-LSTM+CRF+WB (S2S)	**92.96**	98.74	79.66

Table VI.1: Per syllable accuracies of the best data-driven models for the three investigated languages (testing data). The Perceptron is included as a single predictor (the first two rows), Hidden Markov Models and Conditional Random Fields as structured predictors (rows 3-5) and the Neural Network model (last three rows). These methods have been presented in chapter IV and the performed experiments in chapter V.

how the iambic baseline has not been surpassed in the per line results.

The goal of this extrapolation was to check the applicability of common NLP models, trained on poetic data. According to reported results, if tagged data is available, results are promising.

To conclude, empirical results on unsupervised analysis of poetic stress patterns show that the dependencies between outputs help finding structure in raw text. In fact, employing Hidden Markov Models, when the model uses 16 hidden states, results are acceptable. Using smaller models leads to a higher variation among results and larger models fall into a lower accuracy.

Unsupervised models also draw a rough, general structure of the language in question. For example, in English poetry, the distances between stressed syllables is not high.[1] In Spanish poetry, these distances are higher than in English. This is reflected in the unsupervised HMM results. English poems see their result significantly improved when higher order models are used, while improvements in Spanish data are lower, probably because of long dependencies.

[1]The use of structures such as `/ x x x x /` is not common in English poetic tradition.

	English	Spanish	Basque
Perceptron10	28.79	0.44	9.74
Perceptron64	40.86	35.71	8.47
HMM	48.51	45.08	24.10
CRF10	57.28	18.61	26.23
CRF64	51.22	55.44	**26.93**
Bi-LSTM+CRF (W2SP)	44.29	**90.84**	23.75
Bi-LSTM+CRF (S2S)	55.28	63.68	20.32
Bi-LSTM+CRF+WB	**61.39**	88.82	24.67

Table VI.2: Per line accuracies of the best models for the three analyzed languages (testing data).

VI.2 Research questions

At the beginning of this dissertation a set of three research questions were proposed. These questions are addressed below.

Question 1: Which are the informative features when analyzing a poem?

The rule-based scansion system, *ZeuScansion*, shows that simply by using the word's lexical stress and its POS-tag produces positive results. Further, various issues have been checked so as to highlight the importance of the proposed features in this task, and also to extract such information.

- Assess the value of additional features
- Usage of output dependencies
- Bi-LSTMs as feature extractors

In the case of the feature-based predictors, adding additional features that included lexical stresses, POS-tags, syllables and so on, improves significantly the per-syllable accuracies of all feature-based predictors in the English and Spanish data.

If the results obtained by the Averaged Perceptron using 64 features, with its positive results, and the CRFs with 64 features are compared, CRFs show better performance. The improvement is significant and from this, we can infer that the information from output dependencies is crucial when performing scansion, both in English and Spanish.

In the experiments using Deep Learning, we included just syllables or words, with the expectation that the complex model would learn the inherent patterns among characters and syllables or words. Deep Learning experiments gave us better results than previous feature-based models, which means that the most informative feature representation can be extracted by character-level neural models.

Question 2: Does language-specific linguistic knowledge contribute when analyzing poetry?

As mentioned in the previous question, adding additional features that include linguistic information, such as POS-tags and lexical stresses, clearly boosts the accuracies of the classifiers.

In English and Spanish poetry, the DL-based syllable-to-stress models are outperformed by the models that use word boundary information (S2S+WB or W2SP). This shows that the word structure information is helpful for the task of scansion.

Supposing that typical structures among characters would be repeated across languages, a cross-lingual experiment was carried out, as mentioned in section V.7. The low results of this experiment show that character sequences are not repeated among poetic traditions, and thus, language-specific linguistic knowledge helps when analyzing poetry.

Question 3: Is it possible to analyze a poem without having information about language? Can we learn to do it?

Currently, it is possible to reach a per syllable accuracy of around 75% in English and Spanish poetry. Taking into account that this is calculated directly from syllabified text and without resorting to tagged information, results are hopeful. As it may be seen in section VI.4, I expect that including dependencies among lines will be informative for this task.

Beyond current results, the output of unsupervised models seems to be informative as a feature for the task of supervised scansion. As mentioned in chapter III, there are not very many annotated poetry corpora available. Then, semi-supervised approaches could be developed by using these unsupervised models. Unsupervised models could be trained on large unannotated poetic corpora.

VI.3 Contributions

The main contribution of this thesis is the analysis and development of automatic scansion systems with a multilingual perspective. A rule-based system, ZeuScansion, together with numerous data-driven methods were presented.

ZeuScansion scans poems only in English, while the other data-driven methods can analyze poems in more languages, if tagged data are available.

Among data-driven techniques, Neural Network models have shown the best performance. The application of empirical methods showed us which are some of the important aspects when analyzing the meter of poems in different traditions.

Finally, the training and testing of the models have been carried out with three different tagged corpora, in English, Spanish and Basque. The first two were available [Tucker, 2011, Navarro-Colorado, 2015] and the third one was created within this study. This repository includes Basque poems ranging from the 16th century until the 20th century. More information about these corpora can be found in Chapter III.

All the software and resources created in this study have been released under the GNU GPL license, and they are available for download in the following repositories: *ZeuScansion*,[2] *AthenaRhythm*,[3] the Basque poetry corpus,[4] the modified version of the Spanish corpus,[5] and the baselines[6] used for each language.

VI.4 Future work

There are several trends for future that can be derived from the current work. These improvements are proposed for ZeuScansion, feature representations, Neural Network models, corpora and so on.

ZeuScansion calculates the overall meter of the poem, for which the *average stress value* is calculated. This calculation assumes that all lines contain the same number of syllables, which can be a limitation. Tackling this would allow working with poems that contain different numbers of syllables, such as *Phantasmagoria and Other Poems*.

Regarding features, something to consider is the set of basic features. The

[2] `https://github.com/manexagirrezabal/zeuscansion`
[3] `https://github.com/manexagirrezabal/athenarhythm`
[4] `https://bitbucket.org/manexagirrezabal/basquepoetrycorpus`
[5] `https://bitbucket.org/manexagirrezabal/corpussonetossiglosdeoro`
[6] `https://bitbucket.org/manexagirrezabal/baselines4scansion`

applicability of these basic features to other languages should be checked and their language-agnostic nature could be improved.

Currently, all poetic lines are assumed to be independent from each other in all the presented models. This can greatly reduce the accuracy of a scansion system. In order to solve this problem, the inclusion of the global stress information is crucial. Sometimes, when ambiguity cases arise, information of other lines can help disambiguating them. Consider these lines from "*Paradise Lost*", by John Milton:

No more of talk where God or Angel Guest
...
From Eden over Pontus, and the Poole
...
Of Mans First Disobedience, and the Fruit
...
That to the highth of this great Argument

The first two lines' analysis can be done easily, while the third and fourth lines exhibit more complexity. This information can be very useful, especially in the task of unsupervised scansion.

Another interesting element to further explore is rhyme. In the English baselines, there was a high difference between the accuracies of the two iambic baselines. The one that assigned stresses based on the last stress obtained a higher accuracy. The stress pattern of rhyming words must be the same, then, rhyming parts of poems could be helpful toward a better poetic scansion.

Let us move towards the issue of the methods. Nowadays, the best results for poetic analysis have been obtained using Bidirectional LSTMs that model the syllables' character sequence, among more things. The use of bidirectional models is motivated by the fact that LSTMs remember the last visited elements, which means that the forward LSTM remembers the suffixes better and the backward one the prefixes. It is suggested that instead of using a bidirectional LSTM for characters, a tridirectional LSTM could be used. In this way, apart from remembering the beginning and the end of the word/syllable, it also would remember the nucleus of it. So that to do it, the LSTM would go toward the center of the word, as in this example, where the syllable "trans" is modeled:

Forward LSTM: $t \rightarrow r \rightarrow a \rightarrow n \rightarrow s$
Backward LSTM : $t \leftarrow r \leftarrow a \leftarrow n \leftarrow s$
Nucleus LSTM: $t \rightarrow r \rightarrow a \leftarrow n \leftarrow s$

It has been shown that currently the most informative feature representation can be extracted by character-level neural models. This proposes future directions in order to understand what the character level models are actually learning in the poems, as numeric representations are not as understandable as common features (word length, syllable weight and so on).

Recently, new promising Deep Learning models have appeared, and we expect that they will be useful for the current task. These new models include [Kalchbrenner et al., 2016] for supervised poetry scansion or Tran et al. [2016] for unsupervised analysis.

Because of the considerable amount of untagged or raw poetry, unsupervised models should be further explored. In the unsupervised models trained in this study, lines are supposed to be independent, assumption that probably affects the results.

Unsupervised Hidden Markov Models have shown an acceptable performance, and thus, these models should be included in supervised approaches. Using the output of unsupervised Hidden Markov Models as a feature, semi-supervised models could be developed. To this end, the works Daumé III et al. [2010] and Teichert and Daumé III [2009] seem to be promising.

Currently, a two-level representation has been used to evaluate the scansion models, and data-driven models have been trained directly to classify each syllable as stressed or unstressed. This problem can be treated as a multi-class or regression problem, for which the dataset presented in Hayes et al. [2012] would be very useful.

The validity of this work could be demonstrated by getting acoustic information of poems publicly read, using recited poetry corpora. The audios of the poems can be analyzed with software that is currently used for acoustic analysis, such as *Praat* [Boersma et al., 2002].

As the previous goal of this thesis was to automatically generate poetry, I believe that the application of the contributions made in this dissertation will improve previous approaches to poetry generation [Agirrezabal, 2012, Agirrezabal et al., 2013b, Astigarraga et al., 2014].

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016a). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016b). Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*.

Agerri, R., Bermudez, J., and Rigau, G. (2014). Ixa pipeline: Efficient and ready to use multilingual nlp tools. In *LREC*, volume 2014, pages 3823–3828.

Agirrezabal, M. (2012). Bertsobot: lehen urratsak. *Euskal Herriko Unibertsitatea*.

Agirrezabal, M., Alegria, I., Arrieta, B., and Hulden, M. (2012a). BAD: An assistant tool for making verses in Basque. *EACL 2012*, page 13.

Agirrezabal, M., Alegria, I., Arrieta, B., and Hulden, M. (2012b). Finite-state technology in a verse-making tool. In *Proceeding of the 2012 conference on Finite-State Methods and Natural Language Processing*.

Agirrezabal, M., Alegria, I., and Hulden, M. (2012c). Using foma for language-based games. *1st Workshop on Games and NLP (GAMNLP-12), a special session at JapTAL 2012 (the 8th International Conference on Natural Language Processing), Kanazawa, Japan*.

Agirrezabal, M., Alegria, I., and Hulden, M. (2016a). Machine learning for metrical analysis of english poetry. In *Proceedings of COLING 2016,*

the 26th International Conference on Computational Linguistics, Osaka, Japan, pages 772–781.

Agirrezabal, M., Alegria, I., and Hulden, M. (2017). Poesiaren eskantsio automatikoa: bi hizkuntzen azterketa. In *IkerGazte*, volume 1, pages 0–0. UEU.

Agirrezabal, M., Arrieta, B., Astigarraga, A., and Hulden, M. (2013a). Bota bertsoa, eta guk aztertuko dugu: azken urteetako bertsolari txapelketa nagusien analisia. *Elhuyar: zientzia eta teknika*, (300):46–49.

Agirrezabal, M., Arrieta, B., Astigarraga, A., and Hulden, M. (2013b). Postag based poetry generation with wordnet. *Proceedings of the 2013 European Workshop on Natural Language Generation*, page 162.

Agirrezabal, M., Arrieta, B., Astigarraga, A., and Hulden, M. (2013c). ZeuScansion: a tool for scansion of English poetry. *Finite State Methods and Natural Language Processing*, page 18.

Agirrezabal, M., Arrieta, B., Astigarraga, A., and Hulden, M. (2014a). 1986-2013 arteko bertsolari txapelketa nagusien analisi estatistikoa. *Bertsolari Txapelketa Nagusia*.

Agirrezabal, M., Astigarraga, A., Arrieta, B., and Hulden, M. (2016b). Zeuscansion: a tool for scansion of english poetry. *Journal of Language Modelling*, 4(1):3–28.

Agirrezabal, M., Gonzalez-Dios, I., and Lopez-Gazpio, I. (2015). Euskararen sorkuntza automatikoa: lehen urratsak. In *IkerGazte*, volume 1, pages 15–23. UEU.

Agirrezabal, M., Heinz, J., Hulden, M., and Arrieta, B. (2014b). Assigning stress to out-of-vocabulary words: three approaches. *International Conference on Artificial Intelligence, Las Vegas, NV*, 27:105–110.

Aldekoa, I. (1993). *Antología de la poesía vasca = Euskal poesiaren antologia*. Visor, Madrid.

Alegría, I., Arrieta, B., Carreras, X., Díaz de Ilarraza, A., and Uria, L. (2008). Chunk and clause identification for basque by filtering and ranking with perceptrons. *Procesamiento del lenguaje natural. N. 41 (septiembre 2008); pp. 5-12*.

Amenabar, J. (1983). *Euskal poesia kultoaren bilduma, 1880-1963*. Elkar.

Arrieta, B., Alegria, I., and Díaz de Ilarraza, A. (2014). Euskararako komazuzentzaile automatiko baterantz. *EKAIA Euskal Herriko Unibertsitateko Zientzi eta Teknologi Aldizkaria*, (26).

Arzallus, J. (1987). *Antología poética vasca*. Vanguardia Obrera, Madrid.

Astigarraga, A., Agirrezabal, M., Lazkano, E., Jauregi, E., and Sierra, B. (2013). Bertsobot: the first minstrel robot. *6th International Conference on Human System Interaction, Gdansk*.

Astigarraga, A., Jauregi, E., Lazkano, E., and Agirrezabal, M. (2014). Textual coherence in a verse-maker robot. In *Human-Computer Systems Interaction: Backgrounds and Applications 3*, pages 275–287. Springer International Publishing.

Auden, W. H. (1960). The more loving one.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

Baldick, C. (2015). *The Oxford Dictionary of Literary Terms*. Oxford University Press.

Baumann, T. and Meyer-Sickendiek, B. (2016). Large-scale analysis of spoken free-verse poetry. *LT4DH 2016*, page 125.

Bécquer, G. A. (2016). *Rimas y leyendas (Los mejores clásicos)*. PENGUIN CLÁSICOS.

Beesley, K. R. and Karttunen, L. (2003). Finite-state morphology: Xerox tools and techniques. *CSLI, Stanford*.

Bengio, Y. (1999). Markovian models for sequential data. *Neural computing surveys*, 2(1049):129–162.

Berger, J. O. (2013). *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

Bishop, C. M. (2006). Pattern recognition and machine learning. *Machine Learning*, 128.

Blake, W. and Lincoln, A. (1994). *Songs of Innocence and of Experience*, volume 2. Princeton University Press.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

Bobenhausen, K. (2011). The metricalizer2–automated metrical markup of german poetry. *Current Trends in Metrical Analysis, Bern: Peter Lang*, pages 119–131.

Bobenhausen, K. and Hammerich, B. (2015). Métrique littéraire, métrique linguistique et métrique algorithmique de l'allemand mises en jeu dans le programme metricalizer2. *Langages*, (3):67–88.

Boersma, P. P. G. et al. (2002). Praat, a system for doing phonetics by computer. *Glot international*, 5.

Brants, T. (2000). Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Association for Computational Linguistics.

Brogan, T. V. (1981). *English versification, 1570-1980: a reference guide with a global appendix*. Johns Hopkins University Press.

Caparrós, J. D. (1999). *Diccionario de métrica española*. Alianza Editorial.

Carreras, X. (2005). Learning and inference in phrase recognition: A filtering-ranking architecture using perceptron.

Carroll, L. (1869). *Phantasmagoria and Other Poems*. London : Macmillan.

Carroll, L. (1982). Through the looking glass. *The Works of Lewis Carrol, E. Guiliano, ed., Crown Publishers*.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chomsky, N. and Halle, M. (1968). *The sound pattern of English.* Harper & Row, New York.

Coates, C. F. (1998). Charles Baudelaire. Selected Poems From 'Les Fleurs Du Mal': English Renderings and Notes by Norman R. Shapiro: Foreword by Willis Barnstone; Engravings by David Schorr. Chicago and London: The University of Chicago press, 1998. xxxvii+ 209 pp. isbn 0-226-03925-0 (hard).

Collins, M. and Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.

Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

Corn, A. (1997). *The Poem's Heartbeat: A Manual of Prosody.* Copper Canyon Press.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Yarowsky, D., Eisner, J., and Hulden, M. (2016). The SIGMORPHON 2016 shared task—morphological reinflection. *ACL 2016*, page 10.

Daume, H. C. (2006). *Practical structured learning techniques for natural language processing*. PhD thesis.

Daumé III, H. (2012). A course in machine learning. *chapter*, 5:69.

Daumé III, H., Kumar, A., and Saha, A. (2010). Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59. Association for Computational Linguistics.

Delmonte, R. (2016). Exploring shakespeare's sonnets with sparsar. *Linguistics and Literature Studies*, 4(1):61–95.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

DJD Team. Deeplearning4j: Open-source distributed deep learning for the jvm. *Apache Software Foundation License*, 2.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155. ACM.

Durán, R. N. (1982). Esdrújulos inéditos de bartolomé cairasco de figueroa. *Revista de filología de la Universidad de La Laguna*, (1):13–34.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Erro, D., Sainz, I., Saratxaga, I., Navas, E., and Hernáez, I. (2010). Mfcc+ f0 extraction and waveform reconstruction using hnm: preliminary results in an hmm-based synthesizer. *Proc. FALA*, pages 29–32.

Espronceda, J. d. (1999). El estudiante de salamanca.

Estes, A. and Hench, C. (2016). Supervised machine learning for hybrid meter. *on Computational Linguistics for Literature*, page 1.

Fabb, N. and Halle, M. (2008). *Meter in poetry: a new theory*. Cambridge University Press.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874. Software available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

Faruqui, M., Tsvetkov, Y., Neubig, G., and Dyer, C. (2015). Morphological inflection generation using character sequence to sequence learning. *arXiv preprint arXiv:1512.06110*.

Fokkens, A., Soroa, A., Beloki, Z., Ockeloen, N., Rigau, G., van Hage, W. R., and Vossen, P. (2014). Naf and gaf: Linking linguistic annotations. In *Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, pages 9–16.

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

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin.

Fry, S. (2010). *The ode less travelled: Unlocking the poet within*. Random House.

Fussell, P. (1965). *Poetic Meter and Poetic Form*. McGraw Hill.

Gale, W. A., Church, K. W., and Yarowsky, D. (1992). A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5-6):415–439.

Garner, S. R. (1995). Weka: The waikato environment for knowledge analysis. In *Proceedings of the New Zealand computer science research students conference*, pages 57–64. Citeseer.

Garzia, J., Sarasua, J., and Egaña, A. (2001). *The art of bertsolaritza: improvised basque verse singing*. Bertsolari liburuak.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

Gervas, P. (2000). A logic programming application for the analysis of spanish verse. In *Computational Logic—CL 2000*, pages 1330–1344. Springer.

Gervás, P. (2014). Composing narrative discourse for stories of many characters: A case study over a chess game. *Literary and Linguistic Computing*, page fqu040.

Gervás, P., Díaz-Agudo, B., Peinado, F., and Hervás, R. (2005). Story plot generation based on CBR. *Knowledge-Based Systems*, 18(4):235–242.

Gioia, D. (2003). Disappearing ink: Poetry at the end of print culture. *The Hudson Review*, 56(1):21–49.

Goldberg, Y. (2015). A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

González-Blanco García, E. and Rodríguez, J. L. (2013). Remetca: a tei based digital repertory on medieval spanish poetry.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

Gordon, M. (2002). A phonetically driven account of syllable weight. *Language*, pages 51–80.

Gordon, M. (2004). Syllable weight. *Phonetically based phonology*, pages 277–312.

Gorey, E. (1963). *The Gashlycrumb tinies, or, After the outing*. Houghton Mifflin Harcourt.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Greene, E., Bodrumlu, T., and Knight, K. (2010). Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 524–533. Association for Computational Linguistics.

Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

Groves, P. L. (1998). *Strange music: the metre of the English heroic line*, volume 74. English Literary Studies.

Halácsy, P., Kornai, A., and Oravecz, C. (2007). Hunpos: an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 209–212. Association for Computational Linguistics.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Halle, M. and Keyser, S. J. (1971). English stress: Its form, its growth, and its role in verse.

Hanson, K. and Kiparsky, P. (1996). A parametric theory of poetic meter. *Language*, pages 287–335.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Hart, M. (1971). *Project gutenberg*. Project Gutenberg.

Hartlová, E., Intelligentie, B. O. K., and Nack, F. (2013). Mobile social poetry with tweets. *Bachelor thesis, University of Amsterdam*.

Hartman, C. O. (1996). *Virtual muse: experiments in computer poetry*. Wesleyan University Press.

Hartman, C. O. (2005). The Scandroid 1.1.

Hayes, B. (1995). *Metrical stress theory: Principles and case studies*. University of Chicago Press.

Hayes, B. and Wilson, C. (2008). A maximum entropy model of phonotactics and phonotactic learning. *Linguistic inquiry*, 39(3):379–440.

Hayes, B., Wilson, C., and Shisko, A. (2012). Maxent grammars for the metrics of shakespeare and milton. *Language*, 88(4):691–731.

Hayward, M. (1991). A connectionist model of poetic meter. *Poetics*, 20(4):303–317.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification.

Hualde, J. I. (1994). Euskal azentuak eta euskara batua. *Euskera*, 39:1549–1568.

Hulden, M. (2006). Finite-state syllabification. *Finite-State Methods and Natural Language Processing, Lecture Notes in Computer Science*, 4002, 2006:86–96.

Hulden, M. (2009). Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 29–32. Association for Computational Linguistics.

Hulden, M. (2012). Treba: Efficient numerically stable em for pfa. In *ICGI*, pages 249–253.

Igerabide, J. K. (1997). *Gure poesia : antología*. Anaya Haritza, Bilbo.

Isozaki, H. and Kazawa, H. (2002). Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Jacobs, R. A. (1995). Methods for combining experts' probability assessments. *Neural computation*, 7(5):867–888.

Jespersen, O. (1933). Notes on metre. *Linguistica*, pages 249–74.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM.

Jiang, W., Huang, L., Liu, Q., and Lü, Y. (2008). A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. Citeseer.

John, G. H. and Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. v. d., Graves, A., and Kavukcuoglu, K. (2016). Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Kann, K. and Schütze, H. (2016). MED: The LMU system for the SIGMORPHON 2016 Shared Task on Morphological Reinflection. *ACL 2016*, page 62.

Kao, J. and Jurafsky, D. (2012). A computational analysis of style, affect, and imagery in contemporary poetry. In *In Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature. Montreal, Canada*, pages 8–17.

Kaplan, D. M. and Blei, D. M. (2007). A computational approach to style in american poetry. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 553–558. IEEE.

Karpathy, A., Johnson, J., and Fei-Fei, L. (2015). Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Kazama, J., Makino, T., Ohta, Y., and Tsujii, J. (2002). Tuning support vector machines for biomedical named entity recognition. In *Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain-Volume 3*, pages 1–8. Association for Computational Linguistics.

Kraxenberger, M. and Menninghaus, W. (2016). Mimological reveries? disconfirming the hypothesis of phono-emotional iconicity in poetry. *Frontiers in Psychology*, 7.

Kudo, T. and Matsumoto, Y. (2001). Chunking with support vector machines. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

Küper, C. (2012). *Current Trends in Metrical Analysis*. Peter Lang, Bern, Switzerland.

Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lafitte, P. (1967). L'art poétique basque (un inédit d'arnaud d'oyhénart). *Gure Herria*, 39:195–234.

Lafitte, P. and Barbier, J. (1967). *Fantasía y realidad : selección literaria vasca*. Auñamendi, San Sebastian.

Lamb, C., Brown, D. G., and Clarke, C. L. (2016). Evaluating digital poetry: Insights from the cat. In *Proceedings of the Seventh International Conference on Computational Creativity*.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of NAACL-2016, San Diego, California, USA*. Association for Computational Linguistics.

Lekuona, M. d. (1918). La metrica vasca. *Vitoria*, pages 131–157.

Li, Y., Bontcheva, K., and Cunningham, H. (2005). Svm based learning system for information extraction. In *Deterministic and statistical methods in machine learning*, pages 319–339. Springer.

Lilja, E. et al. (2012). Some aspects of poetic rhythm: An essay in cognitive metrics. *Σημειωτκή-Sign Systems Studies*, (1-2):52–64.

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. (2015). Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Logan, H. M. (1988). Computer analysis of sound and meter in poetry. *College Literature*, pages 19–24.

Longfellow, H. W. (1855). *The Song of Hiawatha*. David Bogue.

Lorca, F. G. (2017). *Poesía completa*. Penguin Random House Grupo Editorial España.

Machado, A. and Cano, J. L. (1970). *Campos de castilla*. Taurus Madrid.

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press.

Manurung, R. (2004). An evolutionary algorithm approach to poetry generation.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

McAleese, G. (2007). *Improving Scansion with Syntax: an Investigation into the Effectiveness of a Syntactic Analysis of Poetry by Computer using Phonological Scansion Theory*. PhD thesis, Open University.

McCallum, A. and Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 188–191. Association for Computational Linguistics.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

McCurdy, N., Srikumar, V., and Meyer, M. (2015). Rhymedesign: A tool for analyzing sonic devices in poetry. *Proceedings of Computational Linguistics for Literature*, pages 12–22.

Micoleta, R., de Jesús María, F. J., and Fita, F. (1880). *Modo breve para aprender la lengua vizcayna*. V. Dorca.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent Neural Network based Language Model. In *Interspeech*, volume 2, page 3.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Hlt-naacl*, volume 13, pages 746–751.

Minsky, M. and Papert, S. (1969). Perceptrons.

Mohan, S. P., Soman, K., et al. (2010). Svm based part of speech tagger for malayalam. In *Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference on*, pages 339–341. IEEE.

Monroe, H. (1917). *The new poetry: an anthology*. The Macmillan Company.

Nakagawa, T., Kudo, T., and Matsumoto, Y. (2001). Unknown word guessing and part-of-speech tagging using support vector machines. In *NLPRS*, pages 325–331. Citeseer.

Navarro-Colorado, B. (2015). A computational linguistic approach to spanish golden age sonnets: metrical and semantic aspects. *Computational Linguistics for Literature*, page 105.

Navarro-Colorado, B. (2017). A metrical scansion system for fixed-metre spanish poetry. *Digital Scholarship in the Humanities*.

Navarro-Colorado, B., Lafoz, M. R., and Sánchez, N. (2016). Metrical annotation of a large corpus of spanish sonnets: Representation, scansion and evaluation. In *Proceedings of the Language Resources and Evaluation Conference*.

Navarro-Colorado, B., Ribes-Lafoz, M., and Sánchez, N. (2015). Guía de anotación métrica.

Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Ballesteros, M., Chiang, D., Clothiaux, D., Cohn, T., Duh, K., Faruqui, M., Gan, C., Garrette, D., Ji, Y., Kong, L., Kuncoro, A., Kumar, G., Malaviya, C., Michel, P., Oda, Y., Richardson, M., Saphra, N., Swayamdipta, S., and Yin, P. (2017). Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Nielsen, M. A. (2015). Neural networks and deep learning. *URL: http://neuralnetworksanddeeplearning. com/.(visited: 01.11. 2014)*.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Novak, J., Minematsu, N., and Hirose, K. (2012). Wfst-based Grapheme-to-Phoneme conversion: Open source tools for alignment, model-building and decoding. *Finite-State Methods and Natural Language Processing*, pages 45–49.

Okazaki, N. (2007). Crfsuite: a fast implementation of conditional random fields (crfs).

Oliveira, H. G. (2012). PoeTryMe: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Onaindia, A. (1961). Euskal poesi neurkera. *Olerti*, (2):83–91.

Onaindia, S. (1976). *Mila euskal-olerki eder: bigaŕen argitalpena, ots, lenengoa bera, zerbait aztertu ondoren, facsimil'ez eginda*, volume 2. Editorial La Gran Enciclopedia Vasca.

Ormaechea, N. (1920). De la música y letra popular al acento.

Osinalde, M., Astigarraga, A., Rodriguez, I., and Agirrezabal, M. (2013). Towards basque oral poetry analysis: A machine learning approach. In *RANLP*, pages 119–125.

Otegi, A., Ezeiza, N., Goenaga, I., and Labaka, G. (2016). A modular chain of nlp tools for basque. In *International Conference on Text, Speech, and Dialogue*, pages 93–100. Springer.

Palgrave, F. T. (1914). *The golden treasury of the best songs and lyrical poems in the English language*, volume 133. Oxford University Press.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Perez-de Vinaspre, O., Oronoz, M., Agirrezabal, M., and Lersundi, M. (2013). A finite-state approach to translate snomed ct terms into basque using medical prefixes and suffixes. *Finite State Methods and Natural Language Processing*, 103.

Plamondon, M. R. (2006). Virtual verse analysis: Analysing patterns in poetry. *Literary and Linguistic Computing*, 21(suppl 1):127–141.

Plecháč, P. and Kolár, R. (2015). The corpus of czech verse. *Studia Metrica et Poetica*, 2(1):107–118.

Ponomareva, N., Rosso, P., Pla, F., and Molina, A. (2007). Conditional random fields vs. hidden markov models in a biomedical named entity recognition task. In *Proc. of Int. Conf. Recent Advances in Natural Language Processing, RANLP*, pages 479–483.

Preminger, A., Warnke, F. J., and Hardison Jr, O. B. (2015). *Princeton encyclopedia of poetry and poetics*. Princeton University Press.

Quilis, A. (1967). La percepción de los versos oxítonos, paroxítonos y proparosítonos en español. *Revista de Filología Española*, 50(1):273.

Quilis, A. (1984). *Métrica española*. Ariel Barcelona.

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988a). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. (1988b). *Parallel distributed processing*, volume 1. IEEE.

Ryan, K. M. (2016). Phonological weight. *Language and Linguistics Compass*, 10(12):720–733. LNCO-0669.R1.

Sak, H., Güngör, T., and Saraçlar, M. (2007). Morphological disambiguation of turkish text with perceptron algorithm. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 107–118. Springer.

Sato, K. and Sakakibara, Y. (2005). Rna secondary structural alignment with conditional random fields. *Bioinformatics*, 21(suppl 2):ii237–ii242.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Sejnowski, T. J. and Rosenberg, C. R. (1987). Parallel networks that learn to pronounce English text. *Complex systems*, 1(1):145–168.

Seuss (1953). *Scrambled eggs super!* Random House Books for Young Readers.

Shakespeare, W. (1609). *Shakespeare's sonnets*. Thomas Thorpe.

Shen, L. and Joshi, A. K. (2005). Ranking and reranking with perceptron. *Machine Learning*, 60(1-3):73–96.

Steele, T. (1999). *All the fun's in how you say a thing: an explanation of meter and versification*. Ohio University Press Athens.

Stevens, W. (1923). *Harmonium*. Academy of American Poets.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Sutton, C. and McCallum, A. (2011). An introduction to conditional random fields. *Machine Learning*, 4(4):267–373.

Tarlinskaja, M. (2006). What is 'metricality'? english iambic pentameter. *Formal Approaches to Poetry: Recent Developments in Metrics*, 11:53.

TEI Consortium (2008). *TEI P5: Guidelines for electronic text encoding and interchange*. TEI Consortium.

Teichert, A. R. and Daumé III, H. (2009). Unsupervised part of speech tagging without a lexicon. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*, pages 1–6.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Toivanen, J. M., Toivonen, H., Valitutti, A., et al. (2013). Automatical composition of lyrical songs. In *Proceedings of the Fourth International Conference on Computational Creativity*, page 87.

Tomás, N. T. (1995). *Métrica española*. Ed. Labor.

Tran, K., Bisk, Y., Vaswani, A., Marcu, D., and Knight, K. (2016). Unsupervised neural hidden markov models. *arXiv preprint arXiv:1609.09007*.

Tsur, R. (2008). *Toward a theory of cognitive poetics*. Sussex Academic Press.

Tucker, H. F. (2011). Poetic data and the news from poems: A for better for verse memoir. *Victorian Poetry*, 49(2):267–281.

Urkizu, P. (1994). Oihenarten atsotitzak eta poetika berrirakurriz. *IKER*, 8:295–328.

Urquizu Sarasua, P. (2009). Poesía vasca. antología bilingüe.

Van Merriënboer, B., Bahdanau, D., Dumoulin, V., Serdyuk, D., Warde-Farley, D., Chorowski, J., and Bengio, Y. (2015). Blocks and fuel: Frameworks for deep learning. *arXiv preprint arXiv:1506.00619*.

Veale, T. (2015). Game of tropes: exploring the placebo effect in computational creativity. In *Proceedings of the Sixth International Conference on Computational Creativity June*, page 78.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Weide, R. (1998). The CMU pronunciation dictionary, release 0.6.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Xu, W., Auli, M., and Clark, S. (2015). Ccg supertagging with a recurrent neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 250–255.

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3, pages 195–206.

Young, S. J. and Young, S. (1993). *The HTK hidden Markov model toolkit: Design and philosophy*. University of Cambridge, Department of Engineering.

Zuazo-Zelaieta, K. (1998). Euskalkiak, gaur. *Fontes linguae vasconum: Studia et documenta*, 30(78):191–234.

APPENDIX A

Heuristic for Spanish poetry

In this document we show a heuristic for the transformation of a corpus of Spanish poetry in order to be analyzed using some tools that we developed for English.

Motivation

The interest on this heuristic comes from the impossibility of applying some algorithms developed for the rhythmic analysis of English poetry because of the use of synaloepha in Spanish poetry. This does not mean that this kind of devices are not used in English, as syllable addition is quite typical too. The issue is that in the corpus that we are using for our experiments in English, all syllables are marked with a level of stress, and because of that we decided to mark all the syllables (marking the syllables around synaloephas as unstressed).

Corpus

The corpus we are using for this work **?** has been created and is mantained by Borja Navarro Colorado and it is available on GitHub.[1] This corpus is a collection of sonnets from the Spanish Golden Age and it is composed of 82593 verses, and they include metrical information about each line, calculated automatically using a scansion system. From all the verses, 1921 have been manually checked at this moment (09/06/2016), which will be the ones

[1] `https://github.com/bncolorado/CorpusSonetosSigloDeOro`

that we are interested. We are aplying the heuristic to the manually checked part.

The task

For each line we have to return a stress sequence that will be the allegedly produced sequence when the poem is read aloud. The inputs that we have in this problem are a line of poem, such as,

que o no podréis de lástima escucharme,

along with a sequence of stress values.

- + - + - + - - - + -

By dividing the verse in syllables,

que o no po.dréis de lás.ti.ma es.cu.char.me

we can easily realize that the assignment of the stresses directly to each syllable will leave syllables without assigning any stress value. This is evident because there are 13 syllables in the verse and in the metrical values there are 11 elements (It is a hendecasyllable). This is the input to our heuristic, a verse and a sequence of stresses.

By applying synaloepha to the poem above, the stresses would be assigned in this way:

que_o nO po.drÉis de lÁs.ti.ma_es.cu.chAr.me

The main goal is to add unstressed syllables in a way that the stressed syllables will be kept. Hence, for the above example a possible solution is

- - + - + - + - - - - + -

where each syllable is marked with a level of stress, although, obviously, the time spent to produce the syllables involved in synaloephas is much shorter.

Implementation details

As the corpus of Spanish poetry is not syllabified we had to apply a syllabification algorithm for Spanish written in foma**?**, a finite state toolkit and library, and is mostly based on the syllabification procedure presented in Agirrezabal et al. [2012b], which relies on the maximum onset principle together with the sonority hierarchy. Once the corpus is syllabified, we count the number of syllables and compare it with the length of the metrical tagging.

NOSYLLABLES == NOSTRESSES (589 cases)

If the number of syllables in the line and the number of stresses in the metrical tagging is equal, this means that there is no need of synaloepha, so the metrical values will be returned directly. We have observed 589 cases in our corpus.

NOSYLLABLES != NOSTRESSES (1332 cases)

If the number of syllables and the number of stresses is not the same (1332 verses), some changes must be made:

1. If the number of syllables is higher than the number of stresses: Some syllables must be joined, by using synaloepha (1312 verses).

2. If the number of stresses is higher than the number of syllables: It is a special case that we did not expect (20 verses) and will have to be resolved by hand.

In the first case, we calculate the potential synaloephas that can be made in the verse. Firstly we try to perform synaloepha without including the words that start with the letter 'h' [**?**, p. 26]. If performing all the allowed synaloephas the number of syllables is equal to the metrical syllables, then a solution can be given. This happens in 1166 cases. For example, in the case

no sé ya qué ha.cer.me en mal ta.ma.ño
\+ + + + - + - + - + -

the synaloepha between the syllables *me* and *en* will be chosen and the syllables involving the synaloepha will be unstressed. If making this changes, the number of syllables and the metrical structures length do not coincide, we may need more changes. In such case, synaloephas involving the letter

'h' are performed, which in the corpus there are 146 cases. E.g. in the next example,

en es.te in.cen.dio her.mo.so que par.ti.do
\- + - + - + - - - + -

performing contraction between the syllables *te* and *in* will not suffice because the verse still has 12 syllables. Because of that, we apply synaloephas that include the letter 'h' at the beginnning, so the syllables *dio* and *her* will be joined. The resulting analysis will be this:

en Es.te in.cEn.dio her.mO.so que par.tI.do

Until now, we have seen cases in which making all possible contractions we could assign perfectly the stresses to each syllable. Unfortunately, it is not so straighforward in all the cases. In the verse

y es.tre.cho cuan.do es.tu.vo so.bre mí
\- + - - - + - - - + -

there are 12 syllables and two possible synaloephas that can be made. We could join *y* and *es* or *do* and *es*. In these cases where there are two possible synaloephas, we first generate the possible scansions and evaluate them according to the lexical stress sequence of the verse. We calculate the simple edit distance between the two strings by using only substitution. The lexical stress sequence, the possible scansions and the distances can be seen in table A.1. For the calculation of the lexical stress, we consider accented monosyllabic words as stressed, but the ones that do not have any accent, are tagged with a question mark, indicating that it can be either stressed or unstressed.[2]

Once that we have generated the possible scansions and evaluated them, we choose the one with the minimum edit distance (allowing only substitution) to the lexical stress sequence. If there is not only one minimum but more, we set the verse as ambiguous and accept several scansions, such as in the case,

si no es en ha.ber si.do yo guar.da.do
\- + - - + + - + - + -

[2]We know that there is a weakness in the case that we have a verb like "voy" which is not accented and it should be stressed..

Table A.1: Possible stress sequences for the example in (XXX)

Lexical stresses	?	-	+	-	+	-	-	+	-	+	-	+	
Possibility no. 1	-	+	-	-	-	-	+	-	-	-	+	-	8
Possibility no. 2	-	+	-	-	-	+	-	-	-	-	+	-	8
Possibility no. 3	-	-	+	-	-	-	+	-	-	-	+	-	6
Possibility no. 4	-	-	+	-	-	-	-	+	-	-	-	+	2

where two possible analyses can be got:

- + - - - + + - + - + -
or
- - + - - + + - + - + -

Special cases + error analysis

Although the algorithm seems to be correct in multiple cases, there are still some verses in which it can not give a correct solution. Some weaknesses come from the ambiguity of the syllable division process. For instance, in this poem by Góngora,

Este cíclope, no siciliano,
del microcosmo sí, orbe postrero;
esta antípoda faz, cuyo hemisferio
zona divide en término italiano;

even the rhyming part does not coincide in the rhythmic pattern (first and last lines), which makes (practically) impossible for the machine to calculate correctly the stress structure of the first verse. In the case of the words including the vowel sequence 'ue' or 'ia' it can be considered a dipthong or not, for example, "cru.e.les" and "pue.do". There are some singular cases in which triphthongs must be made, and we have not covered such cases, e.g., "Eres robusto escándalo a orgullosa", where the pronunciation should be "es.cán.da.lo_a_or.gu.llo.sa". We analyzed the results of the heuristic so that to check its correctness and fix errors. The main source of errors was that sometimes a triple vowel synaloepha must be made, for which our heuristic is not prepared. Fortunately, this is not very common in this corpus, so it can be resolved manually. Last but not least, we should mention that we have not found a solution for the 20 verses that will need to be resolved by hand.

At the end, we also realized that our rule-based syllabifier did not perform perfectly, as we expected. It does not divide a word if the left syllable ends with "ns" and the second starts with anything (we have seen the cases like "instante" → "instan.te" and "inscripción" → "inscrip.ción"). We should fix the syllabifier as soon as possible to avoid future errors.

confianza → con.fianza

halagüeña → ha.lagüe.ña (ü is the problem)

Discussion

In this document a heuristic for the modification of a metrically annotated corpus has been presented. The heuristic has some weaknesses, but we think that it can help us in the transformation of the Spanish corpus so that we can apply Machine Learning tools as we did in the case of the corpus in English. Now the main important task is to convert the corpus and manually check the correctness of the generated sequences so that we know the validity of the heuristic.

APPENDIX B

Heuristic for calculating the weight of a syllable in several languages

This function returns 1 if the syllable is heavy and 0 if the syllable is not heavy. It works with a simple heuristic. If the syllable ends in a consonant it will be heavy. If the nucleus of the syllable is a diphthong, it will be heavy too. Otherwise, we will return 0.

Listing B.1: Heuristic to find if a syllable is heavy or not.

```
diphs=[]

def isheavy(str):
    if str[-1] in 'bcdfghjklmnpqrstvxyz': --If the syllable
        ends in consonant, it is heavy
        return '1'
    elif re.sub("[^aeiou]","",str) in diphs: --If the
        nucleus (syllable without consonants) is a diphthong
        , it is heavy
        return '1'
    else: --Then, if neither of the previous conditions are
        satisfied, the syllable is light
        return '0'
```

Diphthongs are calculated from raw text simply by the following algorithm:

Listing B.2: Simple heuristic to extract diphthongs from a text corpus.

```
import sys
import re

f=open(sys.argv[1])
lines = [re.sub("[^a-z\ ]", "", line.decode("utf8").rstrip
    ().lower()) for line in f]
f.close()

def removeChars (w):
  return re.sub("[^aeiou]", "", w)

nuclei=[]
for line in lines:
  nuclei=nuclei+re.split("[^aeiou]", line)

sortedset=sorted([i for i in list(set(nuclei)) if (len(i)<3
     and len(i)>1)])

print sortedset
print ','.join(sortedset)
```