Feedback

Speech Processing, first assignment, November 2014

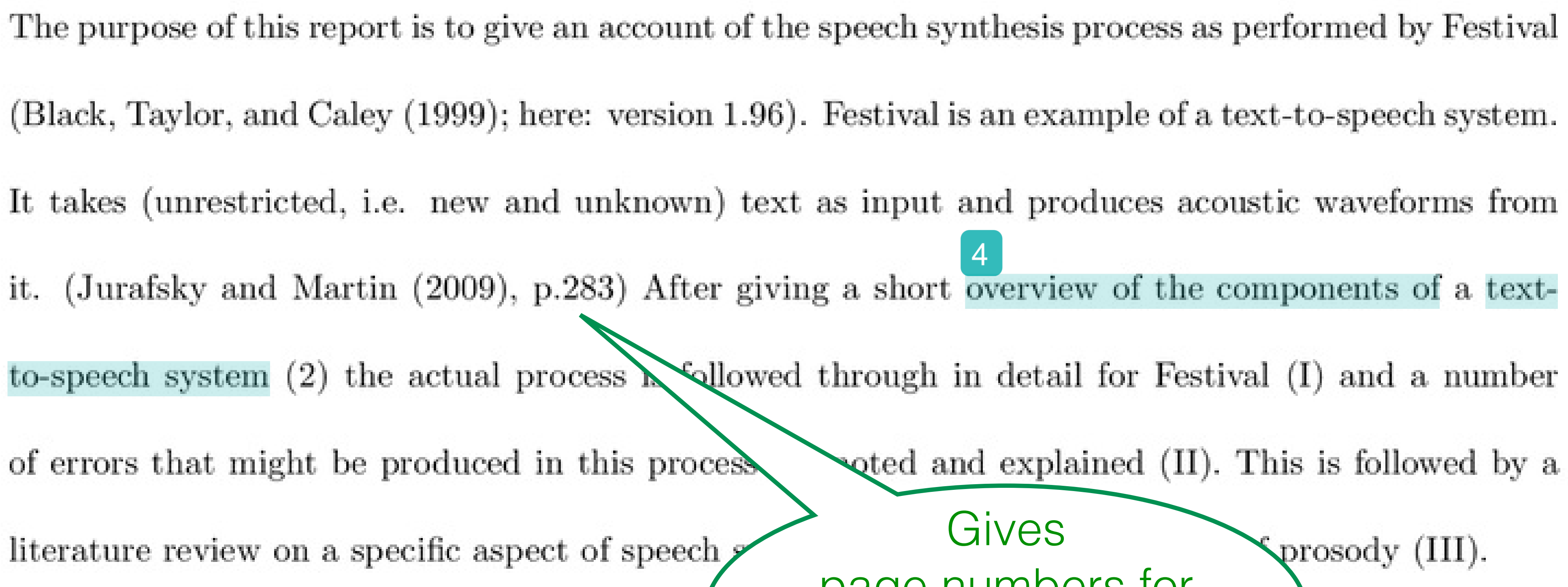
Marking process

- markers were trained by me for this specific assignment, and given examples from a previous year that had been marked by me
    - the lab tutor, Mark, was **not** one of the markers
- Parts I and II
    - UG - marker has taken this course for credit in the past
    - PG - marker was me
- Part III - all UG+PG marked by the same person in (not one of the above)
- all marks were moderated by me & the Part III marker
- marking took ~45 minutes per assignment x 75 assignments

Most common failings

- Simply reporting what happened in the lab when you used Festival
- Failing to demonstrate **understanding of the theory**
  - describing the theory, as you understand it
  - linking practical work to theory
  - using theory to explain why Festival makes mistakes
  - suggesting what would need to be changed to correct these mistakes

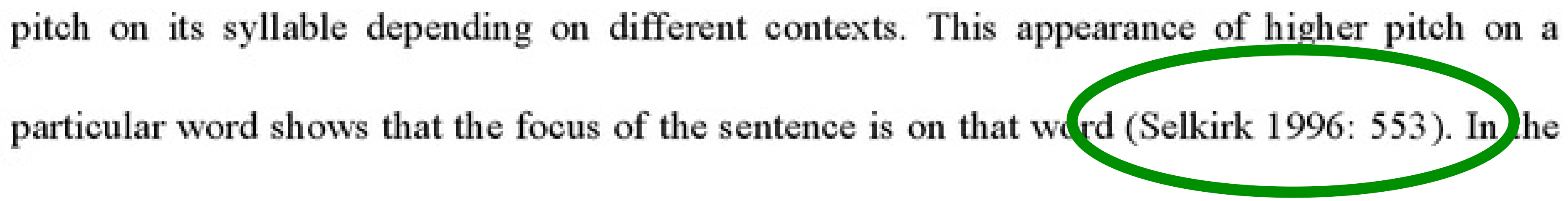
Good citation style: page numbers



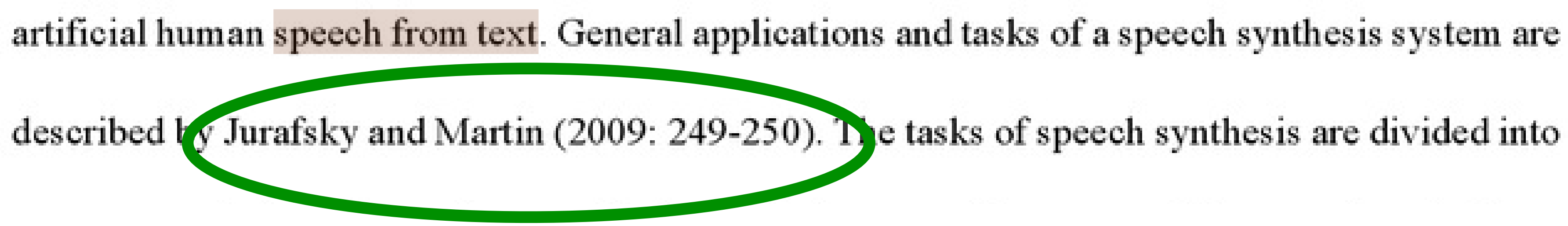
The purpose of this report is to give an account of the speech synthesis process as performed by Festival (Black, Taylor, and Caley (1999); here: version 1.96). Festival is an example of a text-to-speech system. It takes (unrestricted, i.e. new and unknown) text as input and produces acoustic waveforms from it. (Jurafsky and Martin (2009), p.283) After giving a short 4 overview of the components of a text-to-speech system (2) the actual process i followed through in detail for Festival (I) and a number of errors that might be produced in this process oted and explained (II). This is followed by a literature review on a specific aspect of speech s f prosody (III).

Gives page numbers for citations from longer items

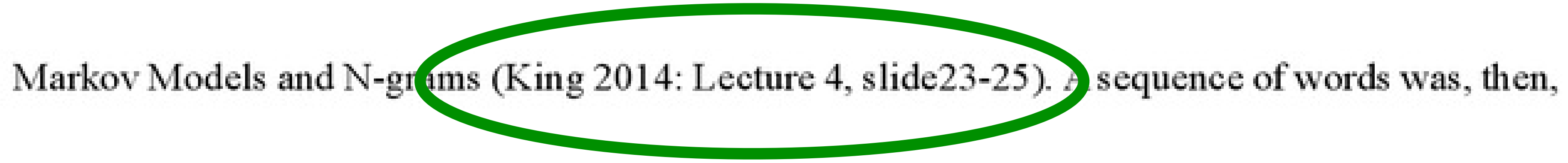
Good citation style: page or slide numbers



pitch on its syllable depending on different contexts. This appearance of higher pitch on a particular word shows that the focus of the sentence is on that word (Selkirk 1996: 553). In the

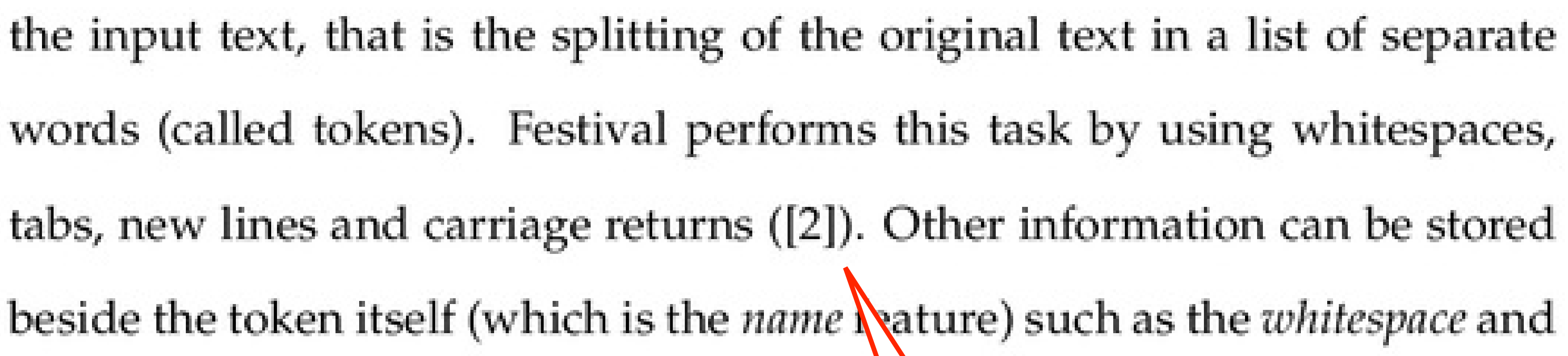


artificial human speech from text. General applications and tasks of a speech synthesis system are described by Jurafsky and Martin (2009: 249-250). The tasks of speech synthesis are divided into



Markov Models and N-grams (King 2014: Lecture 4, slide23-25). A sequence of words was, then,

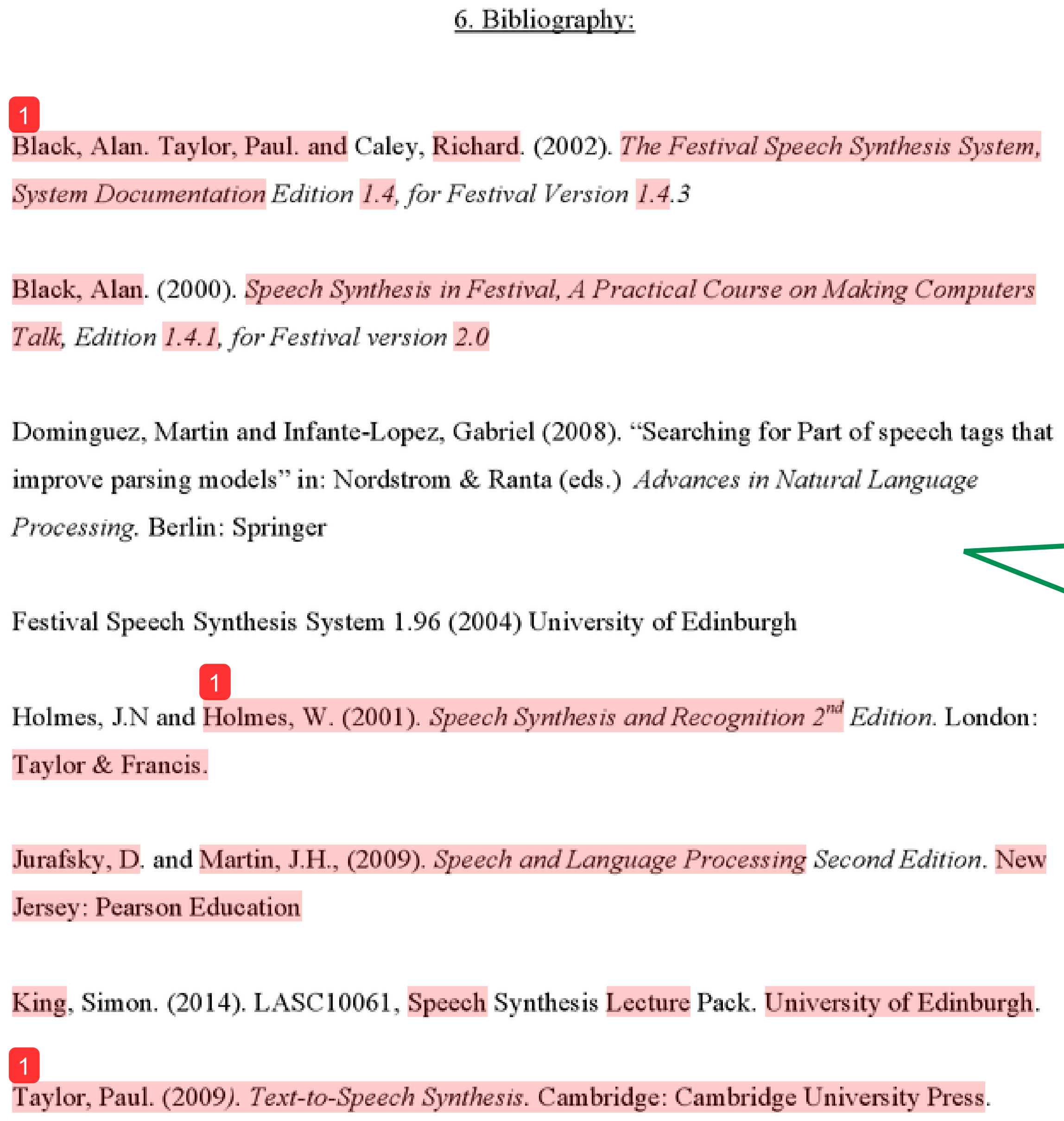
Incorrect citation style



the input text, that is the splitting of the original text in a list of separate words (called tokens). Festival performs this task by using whitespaces, tabs, new lines and carriage returns ([2]). Other information can be stored beside the token itself (which is the *name* feature) such as the *whitespace* and

use […] or (…) but not both

Citing appropriate material



6. Bibliography:

Black, Alan. Taylor, Paul. and Caley, Richard. (2002). *The Festival Speech Synthesis System, System Documentation Edition 1.4, for Festival Version 1.4.3*

Black, Alan. (2000). *Speech Synthesis in Festival, A Practical Course on Making Computers Talk, Edition 1.4.1, for Festival version 2.0*

Dominguez, Martin and Infante-Lopez, Gabriel (2008). "Searching for Part of speech tags that improve parsing models" in: Nordstrom & Ranta (eds.) *Advances in Natural Language Processing*. Berlin: Springer

Festival Speech Synthesis System 1.96 (2004) University of Edinburgh

Holmes, J.N and Holmes, W. (2001). *Speech Synthesis and Recognition 2nd Edition*. London: Taylor & Francis.

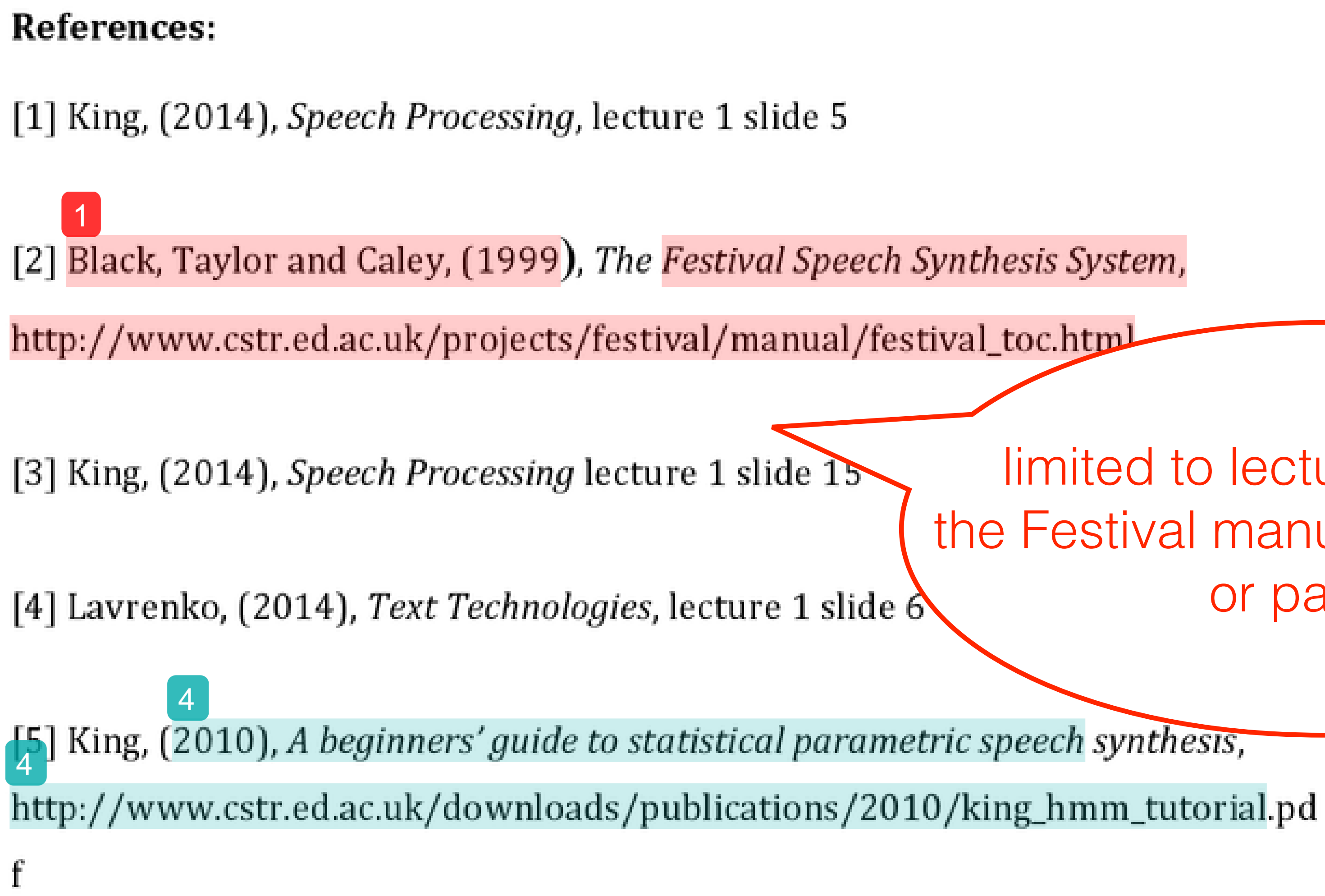
Jurafsky, D. and Martin, J.H., (2009). *Speech and Language Processing Second Edition*. New Jersey: Pearson Education

King, Simon. (2014). LASC10061, Speech Synthesis Lecture Pack. University of Edinburgh.

Taylor, Paul. (2009). *Text-to-Speech Synthesis*. Cambridge: Cambridge University Press.

Includes several items from the set readings, plus an additional item

Citing appropriate material



References:

[1] King, (2014), *Speech Processing*, lecture 1 slide 5

1

[2] Black, Taylor and Caley, (1999), *The Festival Speech Synthesis System*, http://www.cstr.ed.ac.uk/projects/festival/manual/festival_toc.html

[3] King, (2014), *Speech Processing* lecture 1 slide 15

[4] Lavrenko, (2014), *Text Technologies*, lecture 1 slide 6

4

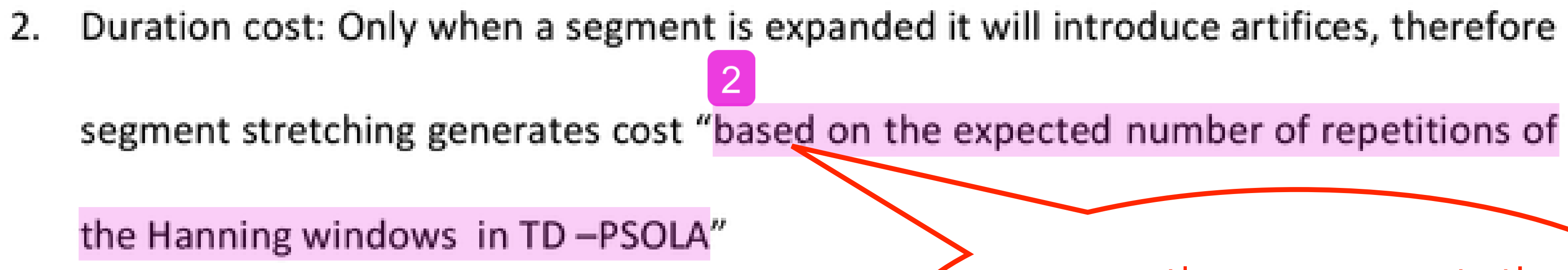
[5] King, (2010), *A beginners' guide to statistical parametric speech synthesis*,

4

http://www.cstr.ed.ac.uk/downloads/publications/2010/king_hmm_tutorial.pd
f

limited to lecture slides and the Festival manual; no textbooks or papers

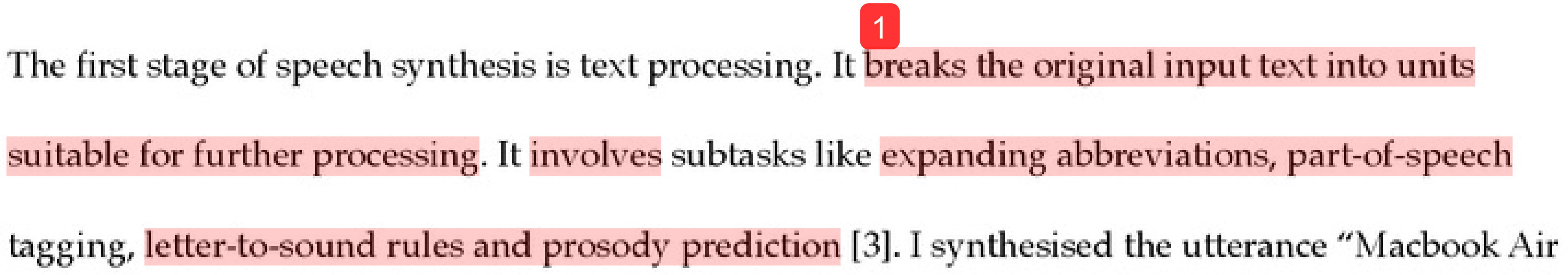
Quoting and attribution



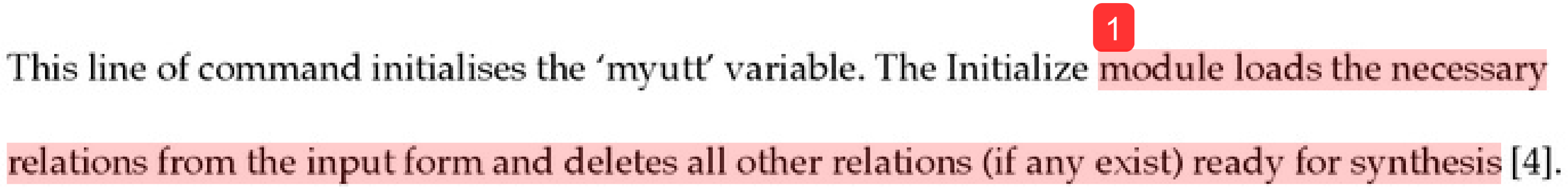
2. Duration cost: Only when a segment is expanded it will introduce artifices, therefore segment stretching generates cost "[2] based on the expected number of repetitions of the Hanning windows in TD –PSOLA"

correctly uses quotation marks, but fails to cite the source of the text

Quoting and attribution



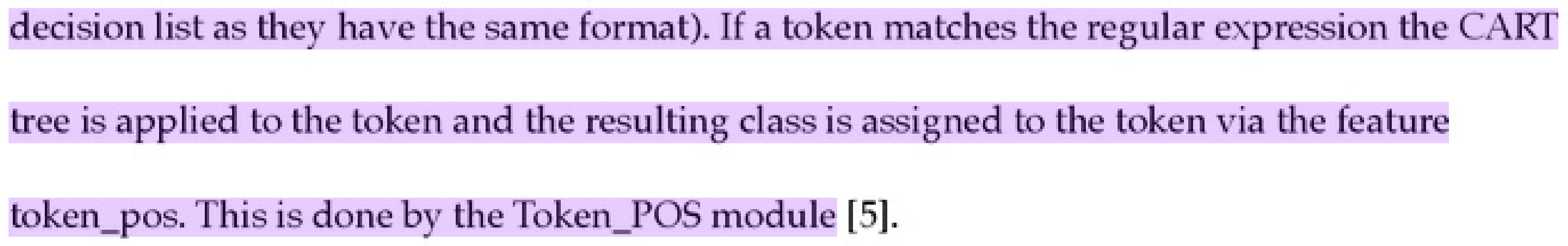
The first stage of speech synthesis is text processing. It breaks the original input text into units suitable for further processing. It involves subtasks like expanding abbreviations, part-of-speech tagging, letter-to-sound rules and prosody prediction [3]. I synthesised the utterance "Macbook Air

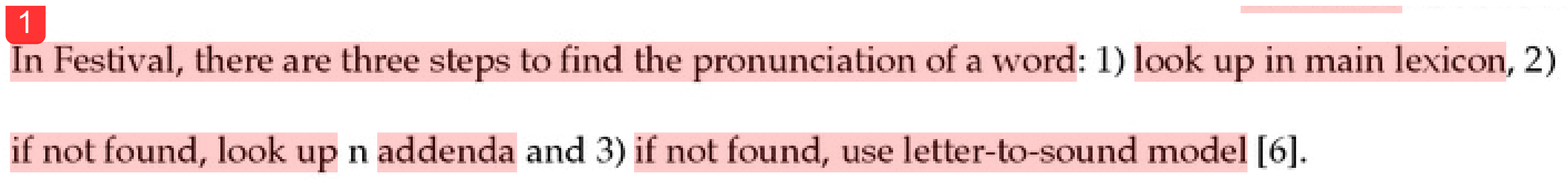


This line of command initialises the 'myutt' variable. The Initialize module loads the necessary relations from the input form and deletes all other relations (if any exist) ready for synthesis [4].

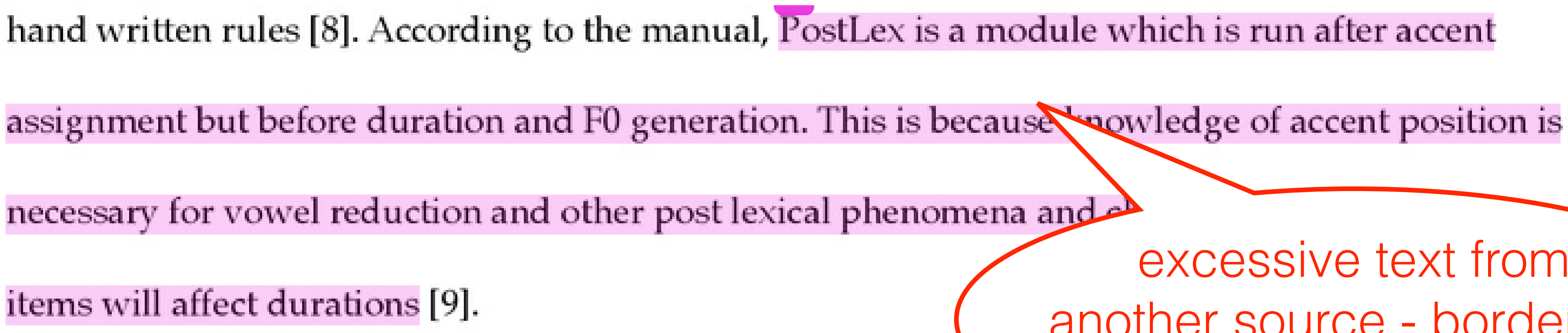
fails to use quotation marks, but does cite the source

Quoting and attribution


decision list as they have the same format). If a token matches the regular expression the CART tree is applied to the token and the resulting class is assigned to the token via the feature token_pos. This is done by the Token_POS module [5].

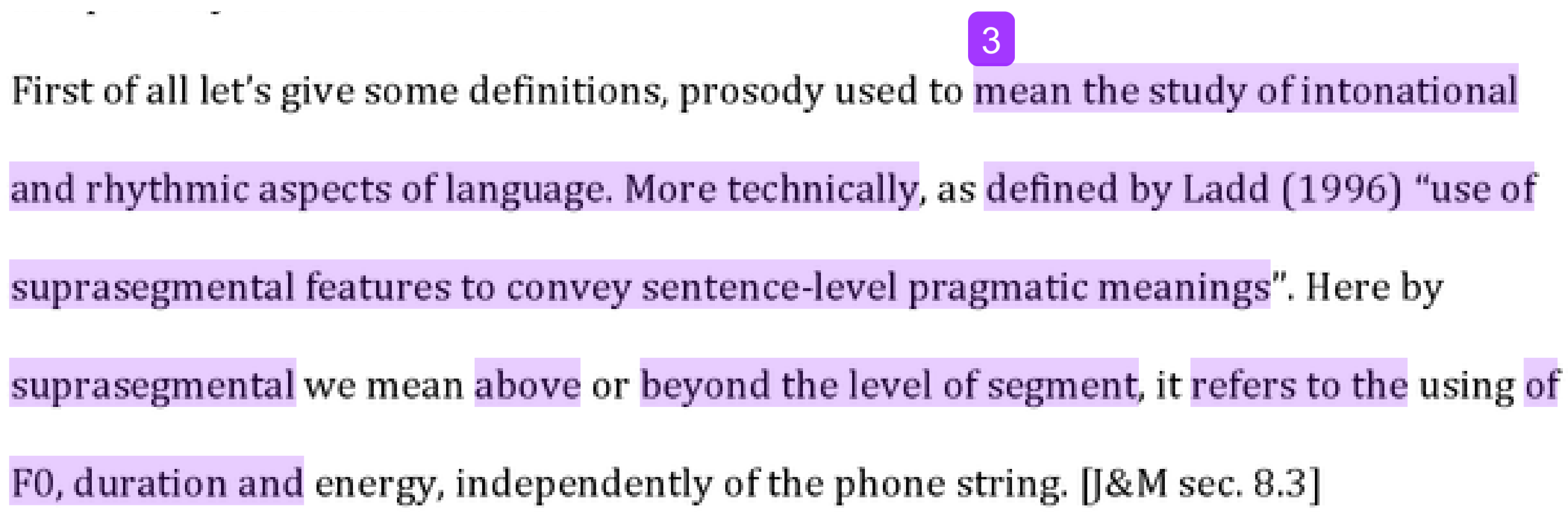

1

In Festival, there are three steps to find the pronunciation of a word: 1) look up in main lexicon, 2) if not found, look up n addenda and 3) if not found, use letter-to-sound model [6].


hand written rules [8]. According to the manual, PostLex is a module which is run after accent assignment but before duration and F0 generation. This is because knowledge of accent position is necessary for vowel reduction and other post lexical phenomena and ot… items will affect durations [9].

excessive text from another source - borderline plagiarism

Quoting and attribution

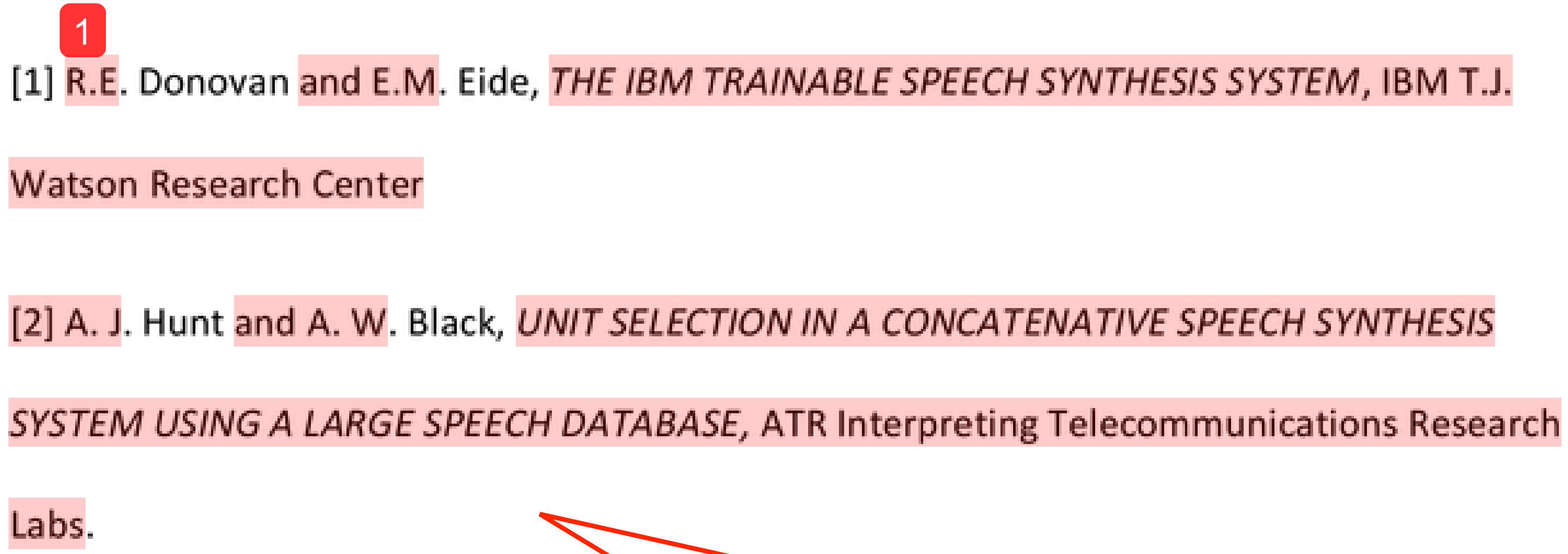


3

First of all let's give some definitions, prosody used to mean the study of intonational and rhythmic aspects of language. More technically, as defined by Ladd (1996) "use of suprasegmental features to convey sentence-level pragmatic meanings". Here by suprasegmental we mean above or beyond the level of segment, it refers to the using of F0, duration and energy, independently of the phone string. [J&M sec. 8.3]

excessive text from another source - borderline plagiarism

Bibliography style

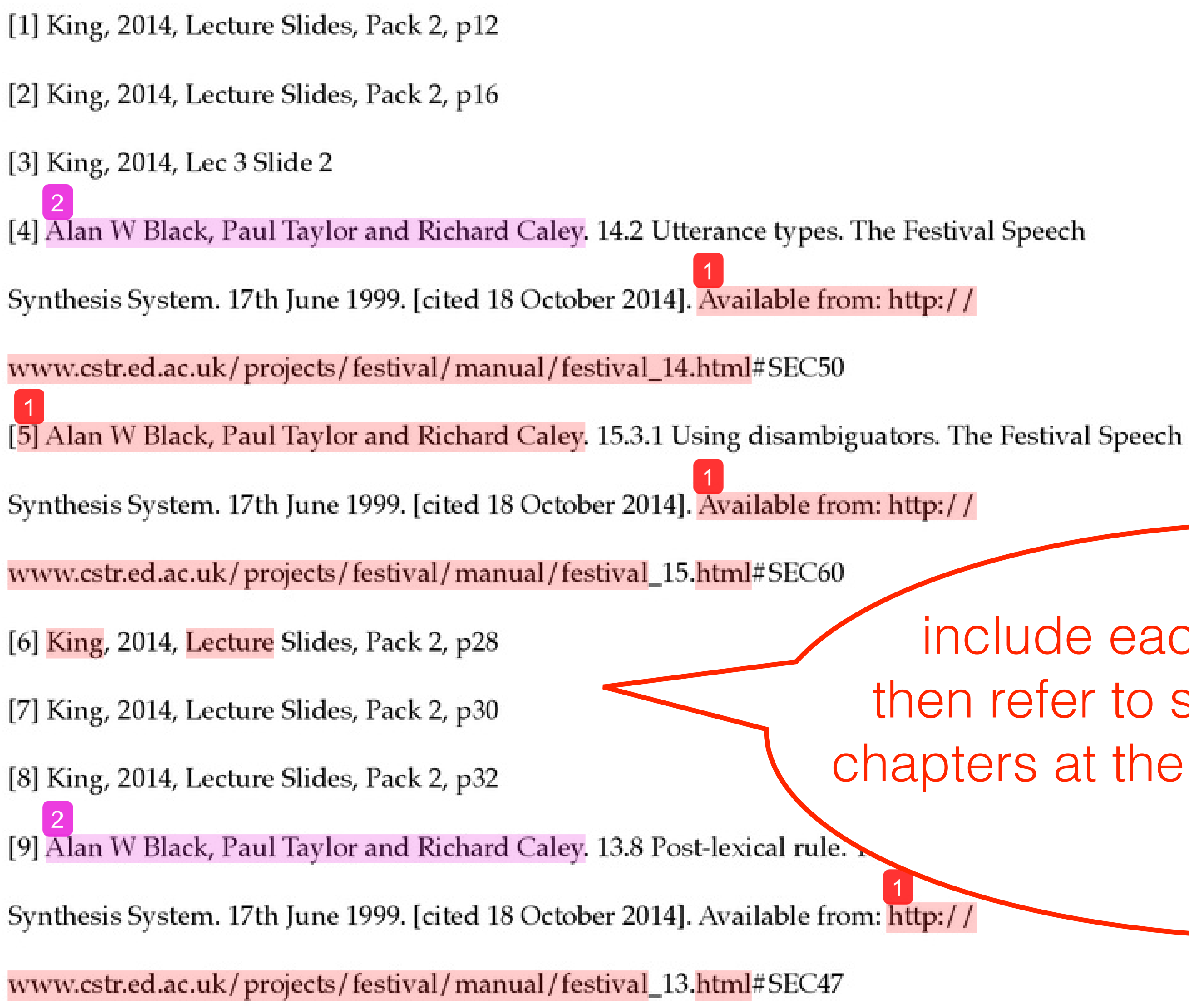

1

[1] R.E. Donovan and E.M. Eide, *THE IBM TRAINABLE SPEECH SYNTHESIS SYSTEM*, IBM T.J. Watson Research Center

[2] A. J. Hunt and A. W. Black, *UNIT SELECTION IN A CONCATENATIVE SPEECH SYNTHESIS SYSTEM USING A LARGE SPEECH DATABASE*, ATR Interpreting Telecommunications Research Labs.

missing names & locations of conferences; no dates; do not need to give the affiliation of the *authors*

Bibliography style



[1] King, 2014, Lecture Slides, Pack 2, p12

[2] King, 2014, Lecture Slides, Pack 2, p16

[3] King, 2014, Lec 3 Slide 2

[4] Alan W Black, Paul Taylor and Richard Caley. 14.2 Utterance types. The Festival Speech Synthesis System. 17th June 1999. [cited 18 October 2014]. Available from: http://www.cstr.ed.ac.uk/projects/festival/manual/festival_14.html#SEC50

[5] Alan W Black, Paul Taylor and Richard Caley. 15.3.1 Using disambiguators. The Festival Speech Synthesis System. 17th June 1999. [cited 18 October 2014]. Available from: http://www.cstr.ed.ac.uk/projects/festival/manual/festival_15.html#SEC60

[6] King, 2014, Lecture Slides, Pack 2, p28

[7] King, 2014, Lecture Slides, Pack 2, p30

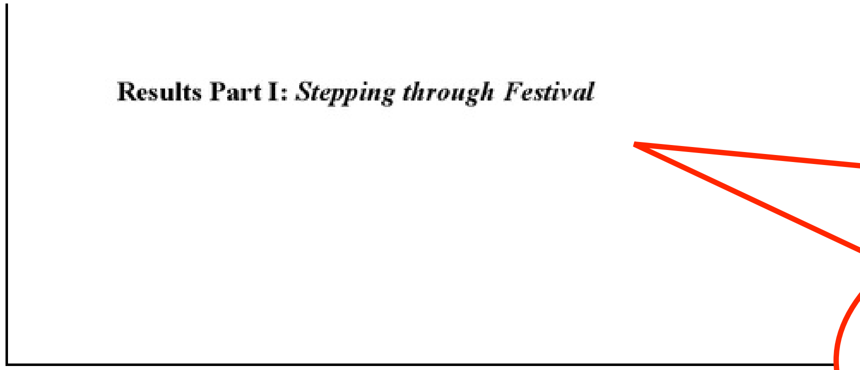
[8] King, 2014, Lecture Slides, Pack 2, p32

[9] Alan W Black, Paul Taylor and Richard Caley. 13.8 Post-lexical rule.

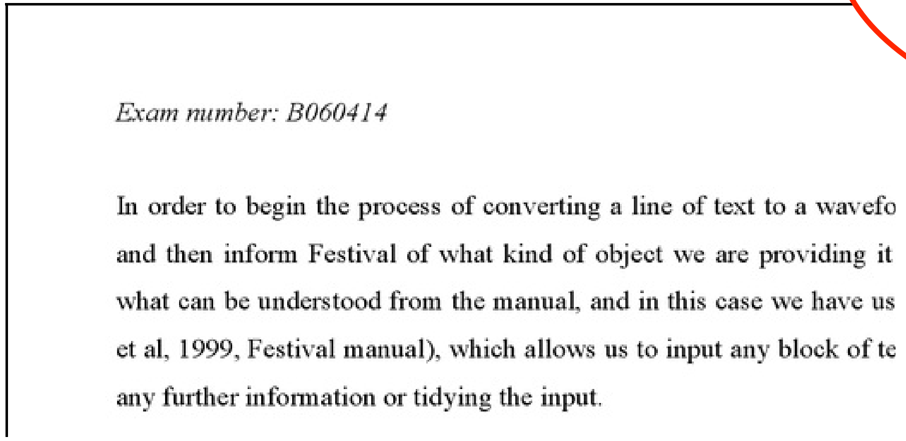
Synthesis System. 17th June 1999. [cited 18 October 2014]. Available from: http://www.cstr.ed.ac.uk/projects/festival/manual/festival_13.html#SEC47

include each item just **once** and then refer to specific pages / slides / chapters at the point where you cite it in the text

Formatting



Results Part I: ***Stepping through Festival***

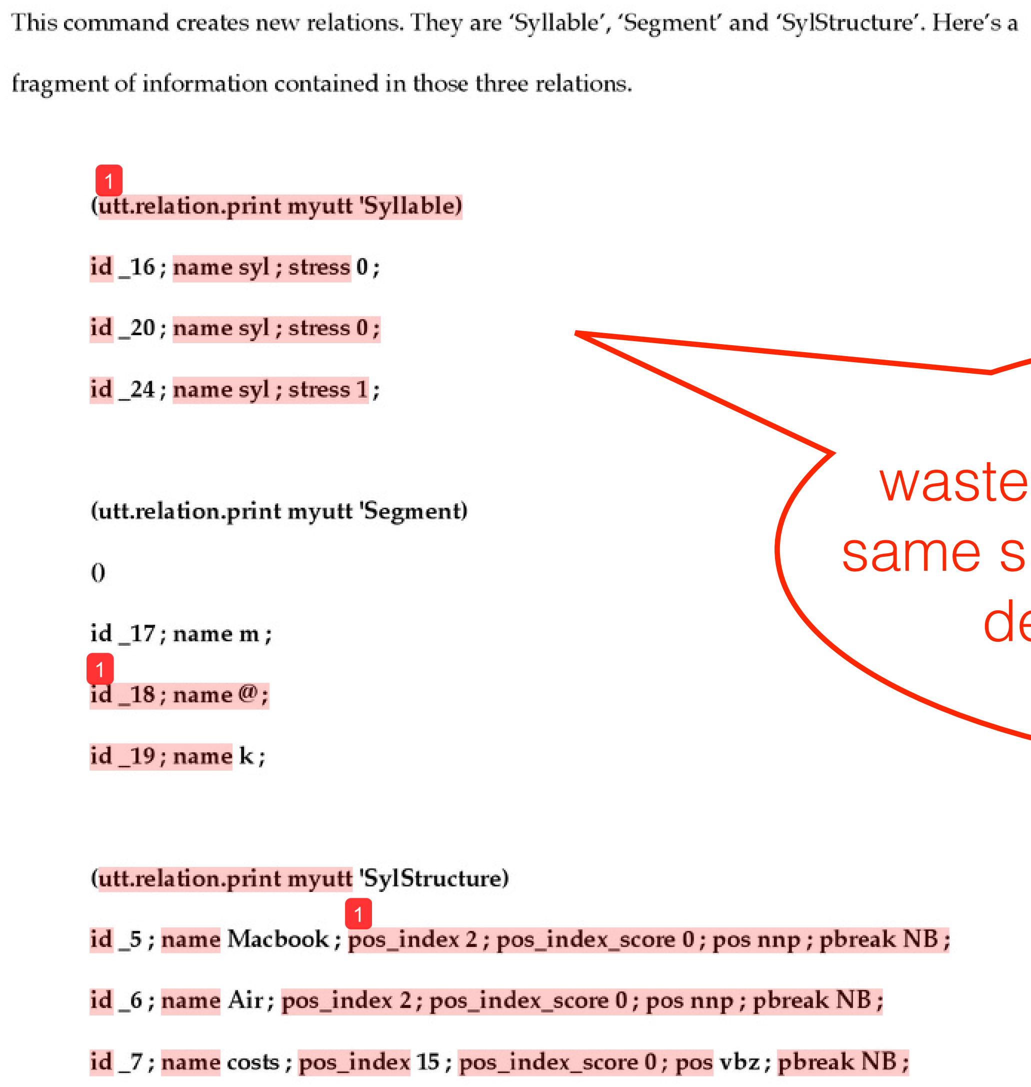


Exam number: B060414

In order to begin the process of converting a line of text to a wavefo
and then inform Festival of what kind of object we are providing it
what can be understood from the manual, and in this case we have us
et al, 1999, Festival manual), which allows us to input any block of te
any further information or tidying the input.

avoid “widows” & “orphans” - single lines (**especially headings**) on their own

Formatting

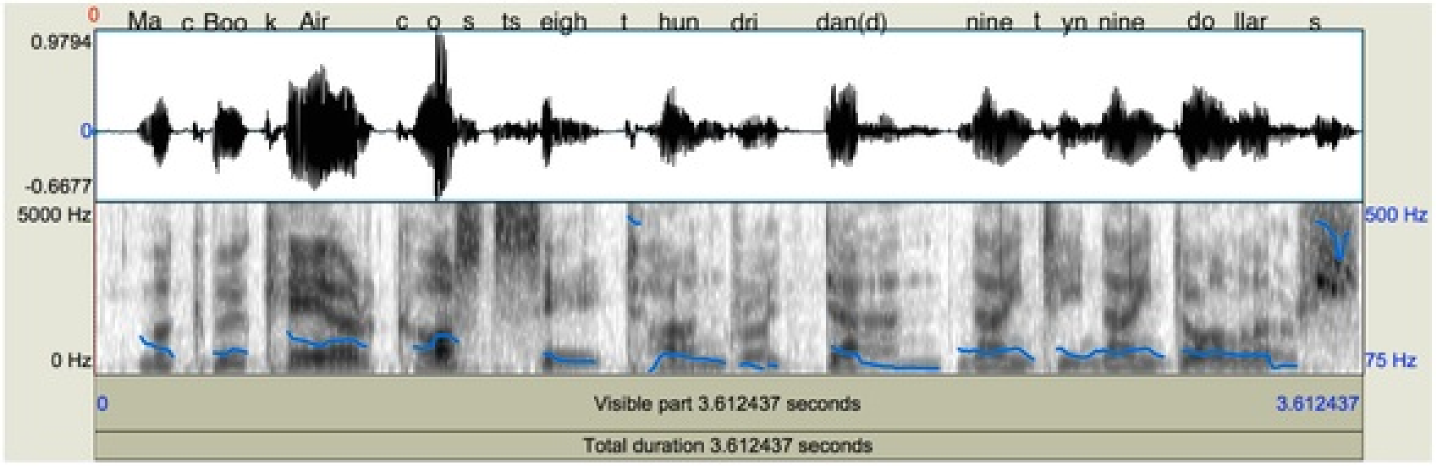


This command creates new relations. They are ‘Syllable’, ‘Segment’ and ‘SylStructure’. Here’s a fragment of information contained in those three relations.

```
(utt.relation.print myutt 'Syllable)

id _16 ; name syl ; stress 0 ;

id _20 ; name syl ; stress 0 ;

id _24 ; name syl ; stress 1 ;


(utt.relation.print myutt 'Segment)

()

id _17 ; name m ;

id _18 ; name @ ;

id _19 ; name k ;


(utt.relation.print myutt 'SylStructure)

id _5 ; name Macbook ; pos_index 2 ; pos_index_score 0 ; pos nnp ; pbreak NB ;

id _6 ; name Air ; pos_index 2 ; pos_index_score 0 ; pos nnp ; pbreak NB ;

id _7 ; name costs ; pos_index 15 ; pos_index_score 0 ; pos vbz ; pbreak NB ;
```

waste of space - could have used the same space to gain a **lot** more marks by demonstrating understanding

Formatting

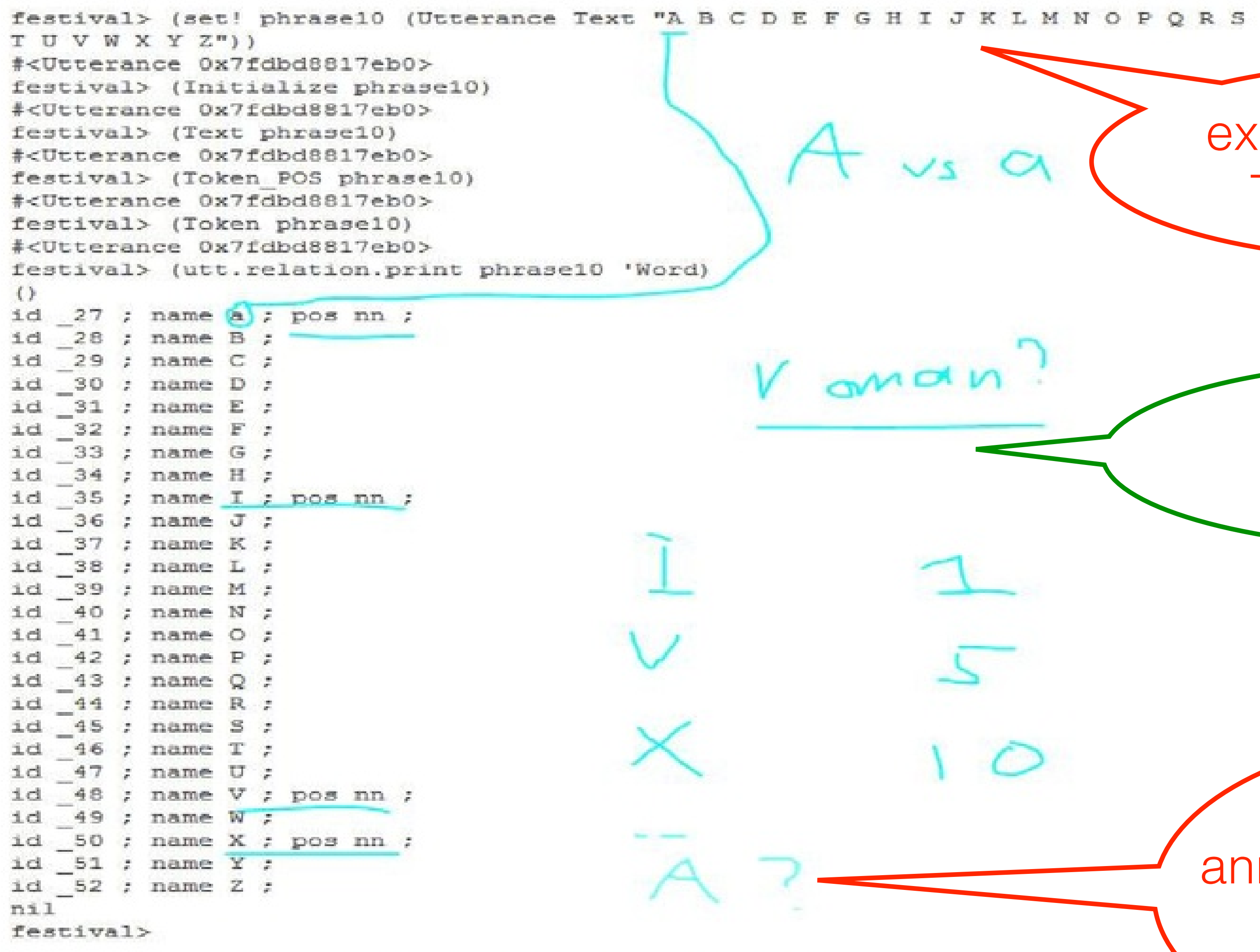


the following 'and'. So it sounds like 'dan' is a separate word rather than a smooth transition.

Figure 1. The waveform of 'Macbook Air costs $899 dollars' with rough mapping with actual spelling

caption must go directly below (or above) the figure with no intervening text

Formatting



```
festival> (set! phrase10 (Utterance Text "A B C D E F G H I J K L M N O P Q R S
T U V W X Y Z"))
#<Utterance 0x7fdbd8817eb0>
festival> (Initialize phrase10)
#<Utterance 0x7fdbd8817eb0>
festival> (Text phrase10)
#<Utterance 0x7fdbd8817eb0>
festival> (Token_POS phrase10)
#<Utterance 0x7fdbd8817eb0>
festival> (Token phrase10)
#<Utterance 0x7fdbd8817eb0>
festival> (utt.relation.print phrase10 'Word)
()
id _27 ; name a ; pos nn ;
id _28 ; name B ;
id _29 ; name C ;
id _30 ; name D ;
id _31 ; name E ;
id _32 ; name F ;
id _33 ; name G ;
id _34 ; name H ;
id _35 ; name I ; pos nn ;
id _36 ; name J ;
id _37 ; name K ;
id _38 ; name L ;
id _39 ; name M ;
id _40 ; name N ;
id _41 ; name O ;
id _42 ; name P ;
id _43 ; name Q ;
id _44 ; name R ;
id _45 ; name S ;
id _46 ; name T ;
id _47 ; name U ;
id _48 ; name V ; pos nn ;
id _49 ; name W ;
id _50 ; name X ; pos nn ;
id _51 ; name Y ;
id _52 ; name Z ;
nil
festival>
```

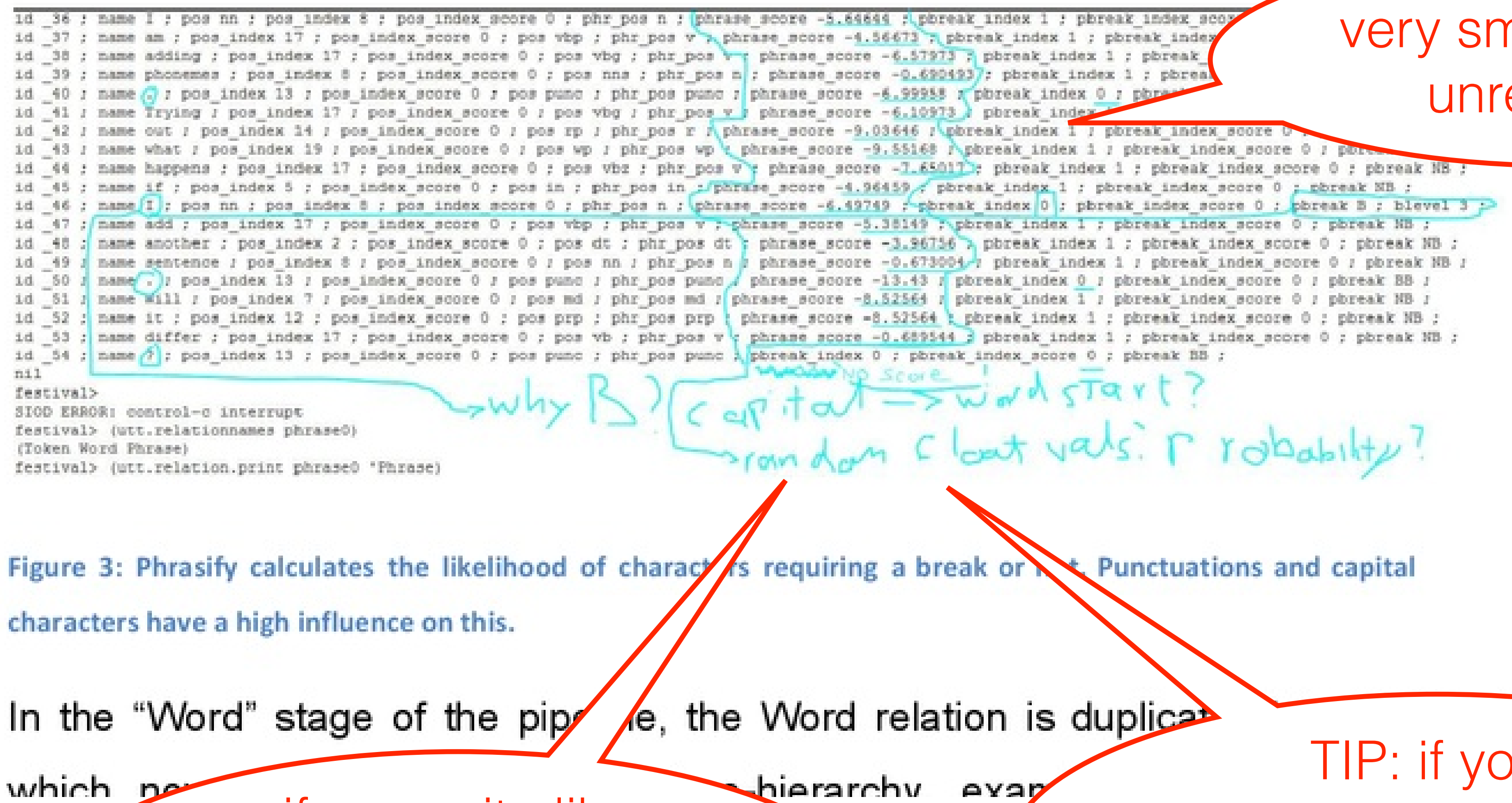
Figure 1: Specific capital characters when separated receive a pos-tag in the Token-stage.

excessive verbatim Terminal output

but at least it is annotated

except that the annotation is cryptic and it's not clear what it means

Formatting



```
id _36 ; name I ; pos nn ; pos_index 8 ; pos_index_score 0 ; phr_pos n ; phrase_score -5.64644 ; pbreak_index 1 ; pbreak_index_scor
id _37 ; name am ; pos_index 17 ; pos_index_score 0 ; pos vbp ; phr_pos v ; phrase_score -4.56673 ; pbreak_index 1 ; pbreak_index
id _38 ; name adding ; pos_index 17 ; pos_index_score 0 ; pos vbg ; phr_pos v ; phrase_score -6.57973 ; pbreak_index 1 ; pbreak_
id _39 ; name phonemes ; pos_index 8 ; pos_index_score 0 ; pos nns ; phr_pos n ; phrase_score -0.690493 ; pbreak_index 1 ; pbrea
id _40 ; name . ; pos_index 13 ; pos_index_score 0 ; pos punc ; phr_pos punc ; phrase_score -6.99958 ; pbreak_index 0 ; pbre
id _41 ; name Trying ; pos_index 17 ; pos_index_score 0 ; pos vbg ; phr_pos v ; phrase_score -6.10973 ; pbreak_index
id _42 ; name out ; pos_index 14 ; pos_index_score 0 ; pos rp ; phr_pos r ; phrase_score -9.03646 ; pbreak_index 1 ; pbreak_index_score 0
id _43 ; name what ; pos_index 19 ; pos_index_score 0 ; pos wp ; phr_pos wp ; phrase_score -9.55168 ; pbreak_index 1 ; pbreak_index_score 0 ; pbre
id _44 ; name happens ; pos_index 17 ; pos_index_score 0 ; pos vbz ; phr_pos v ; phrase_score -7.65017 ; pbreak_index 1 ; pbreak_index_score 0 ; pbreak NB ;
id _45 ; name if ; pos_index 5 ; pos_index_score 0 ; pos in ; phr_pos in ; phrase_score -4.96459 ; pbreak_index 1 ; pbreak_index_score 0 ; pbreak NB ;
id _46 ; name I ; pos nn ; pos_index 8 ; pos_index_score 0 ; phr_pos n ; phrase_score -6.49749 ; pbreak_index 0 ; pbreak_index_score 0 ; pbreak B ; blevel 3 ;
id _47 ; name add ; pos_index 17 ; pos_index_score 0 ; pos vbp ; phr_pos v ; phrase_score -5.38149 ; pbreak_index 1 ; pbreak_index_score 0 ; pbreak NB ;
id _48 ; name another ; pos_index 2 ; pos_index_score 0 ; pos dt ; phr_pos dt ; phrase_score -3.96736 ; pbreak_index 1 ; pbreak_index_score 0 ; pbreak NB ;
id _49 ; name sentence ; pos_index 8 ; pos_index_score 0 ; pos nn ; phr_pos n ; phrase_score -0.673004 ; pbreak_index 1 ; pbreak_index_score 0 ; pbreak NB ;
id _50 ; name . ; pos_index 13 ; pos_index_score 0 ; pos punc ; phr_pos punc ; phrase_score -13.43 ; pbreak_index 0 ; pbreak_index_score 0 ; pbreak BB ;
id _51 ; name will ; pos_index 7 ; pos_index_score 0 ; pos md ; phr_pos md ; phrase_score -8.52564 ; pbreak_index 1 ; pbreak_index_score 0 ; pbreak NB ;
id _52 ; name it ; pos_index 12 ; pos_index_score 0 ; pos prp ; phr_pos prp ; phrase_score -8.52564 ; pbreak_index 1 ; pbreak_index_score 0 ; pbreak NB ;
id _53 ; name differ ; pos_index 17 ; pos_index_score 0 ; pos vb ; phr_pos v ; phrase_score -0.689544 ; pbreak_index 1 ; pbreak_index_score 0 ; pbreak NB ;
id _54 ; name ? ; pos_index 13 ; pos_index_score 0 ; pos punc ; phr_pos punc ; pbreak_index 0 ; pbreak_index_score 0 ; pbreak BB ;
nil
festival>
SIOD ERROR: control-c interrupt
festival> (utt.relationnames phrase0)
(Token Word Phrase)
festival> (utt.relation.print phrase0 'Phrase)
```

Figure 3: Phrasify calculates the likelihood of charact rs requiring a break or t. Punctuations and capital characters have a high influence on this.

In the “Word” stage of the pip e, the Word relation is duplicat

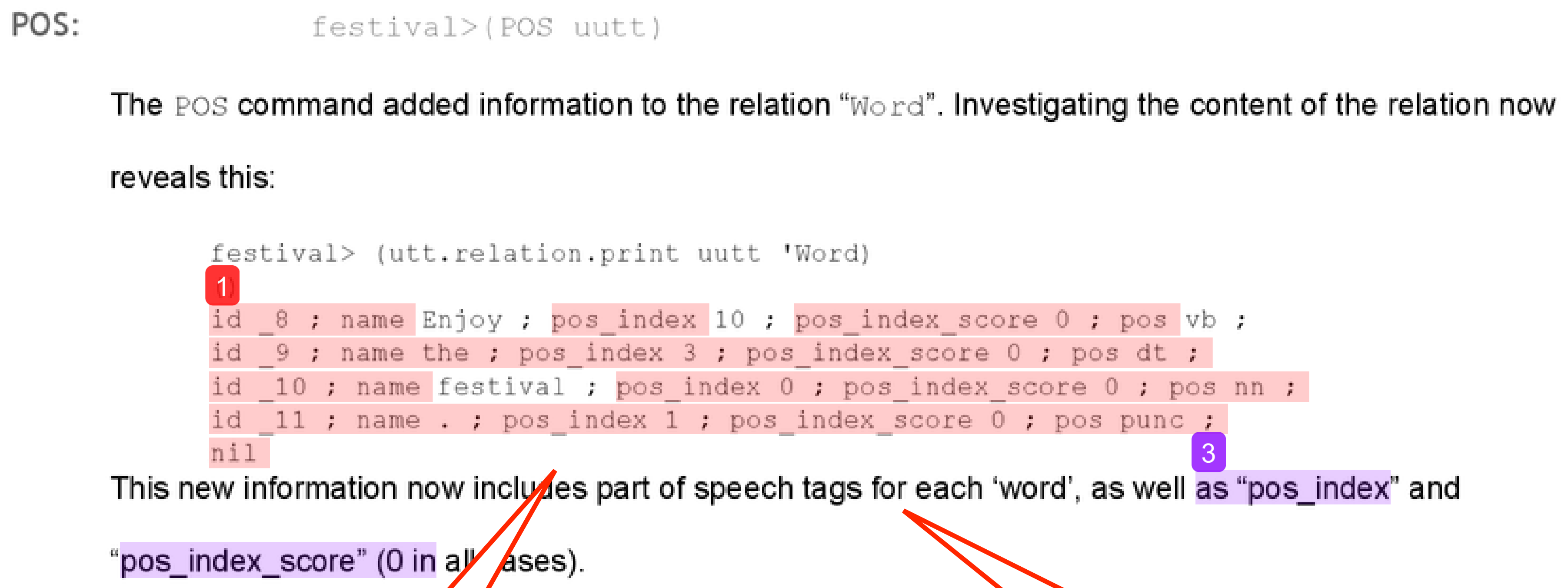
which ne hierarchy exam

very small - almost unreadable

if you write like this in an exam, the marker may not be able to read it

TIP: if your handwriting is not clear, then type your annotations

Insufficient detail



POS: festival>(POS uutt)

The POS command added information to the relation "Word". Investigating the content of the relation now reveals this:

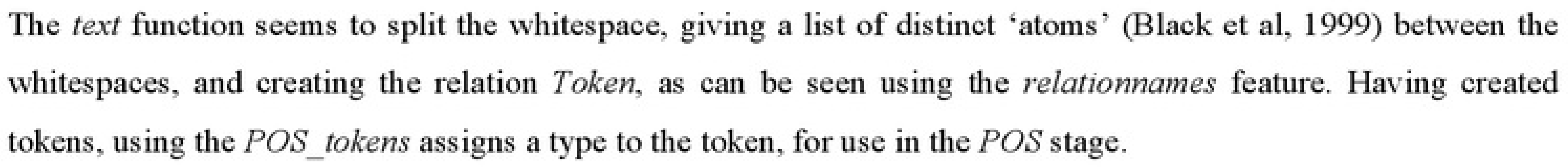
```
festival> (utt.relation.print uutt 'Word)
()
id _8 ; name Enjoy ; pos_index 10 ; pos_index_score 0 ; pos vb ;
id _9 ; name the ; pos_index 3 ; pos_index_score 0 ; pos dt ;
id _10 ; name festival ; pos_index 0 ; pos_index_score 0 ; pos nn ;
id _11 ; name . ; pos_index 1 ; pos_index_score 0 ; pos punc ;
nil
```

This new information now includes part of speech tags for each 'word', as well as "pos_index" and "pos_index_score" (0 in all cases).

simply reports Festival output

doesn't explain WHY tagging is necessary or HOW it is done

Insufficient detail

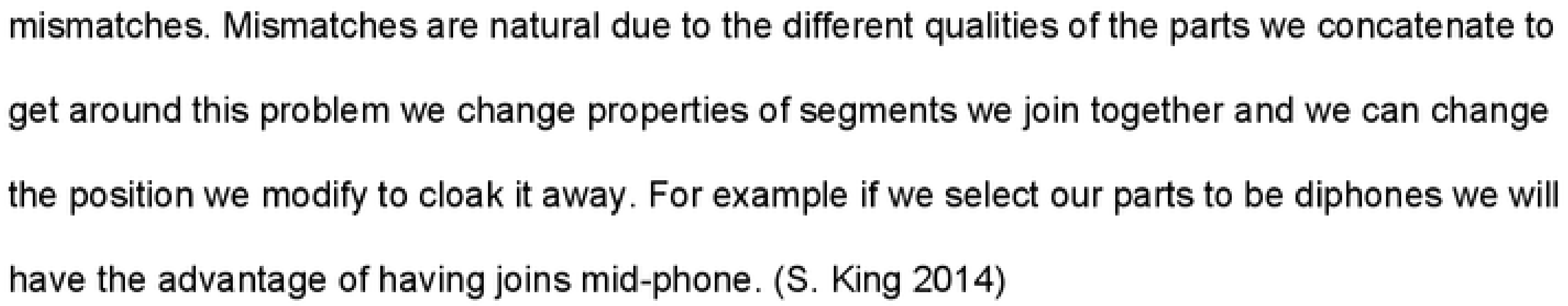


The *text* function seems to split the whitespace, giving a list of distinct ‘atoms’ (Black et al, 1999) between the whitespaces, and creating the relation *Token*, as can be seen using the *relationnames* feature. Having created tokens, using the *POS_tokens* assigns a type to the token, for use in the *POS* stage.

doesn’t give any detail on HOW any of this is done

doesn’t fully explain WHY any of this is necessary

Insufficient detail



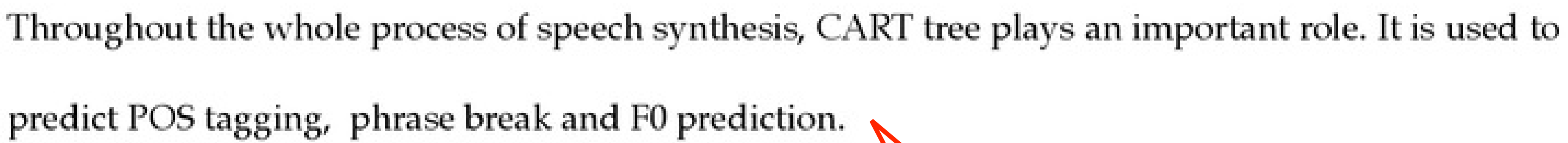
mismatches. Mismatches are natural due to the different qualities of the parts we concatenate to get around this problem we change properties of segments we join together and we can change the position we modify to cloak it away. For example if we select our parts to be diphones we will have the advantage of having joins mid-phone. (S. King 2014)

nothing incorrect here

but a little more detail would get a lot more marks: what precisely is this mismatch?

Insufficient detail

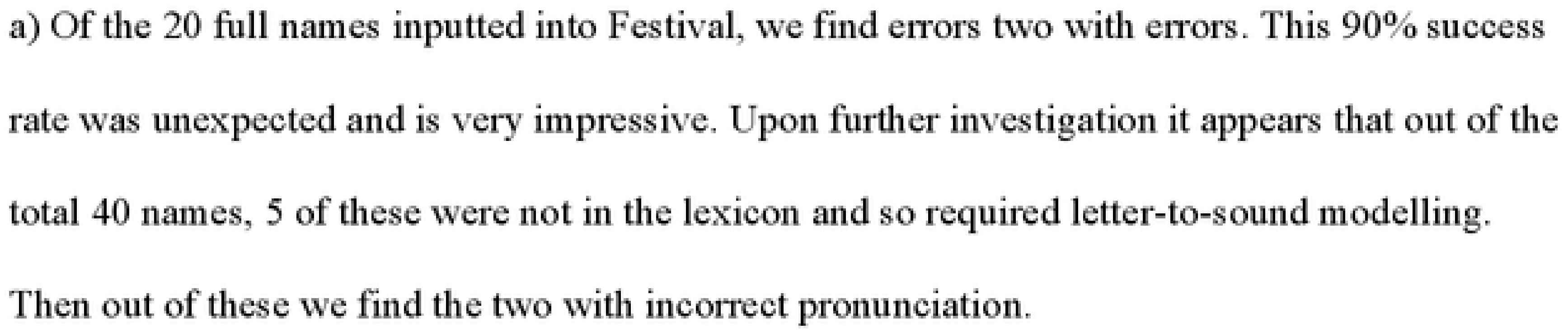
yes, Festival uses CARTs for several tasks



Throughout the whole process of speech synthesis, CART tree plays an important role. It is used to predict POS tagging, phrase break and F0 prediction.

but how are they used? what are the predictors and predictees? what data are they trained on ?

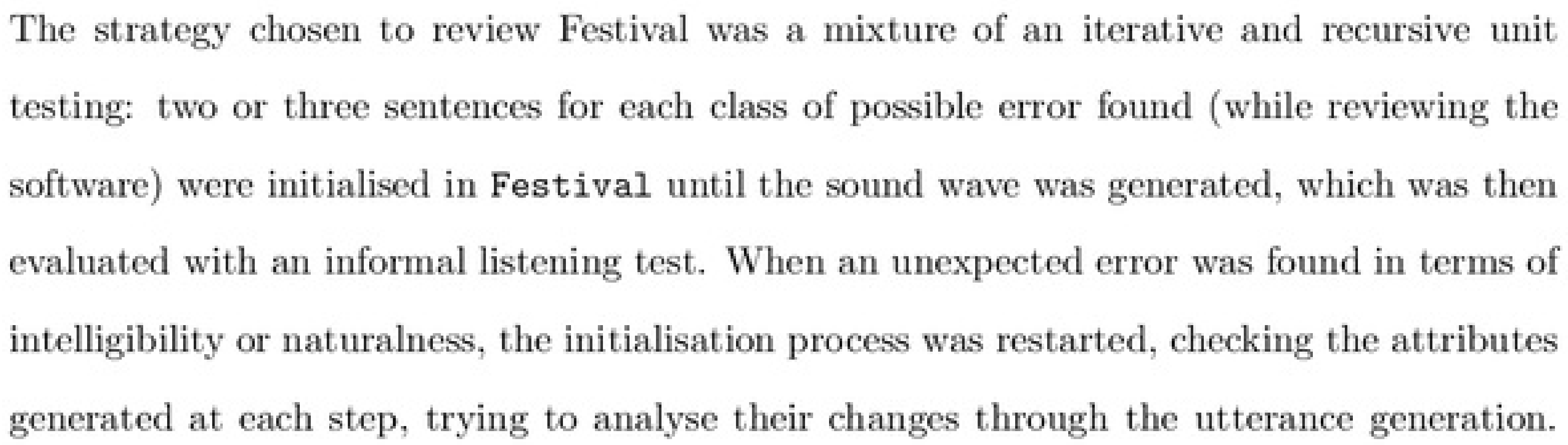
Methodology



a) Of the 20 full names inputted into Festival, we find errors two with errors. This 90% success rate was unexpected and is very impressive. Upon further investigation it appears that out of the total 40 names, 5 of these were not in the lexicon and so required letter-to-sound modelling. Then out of these we find the two with incorrect pronunciation.

systematic, considers multiple examples and measures accuracy rate

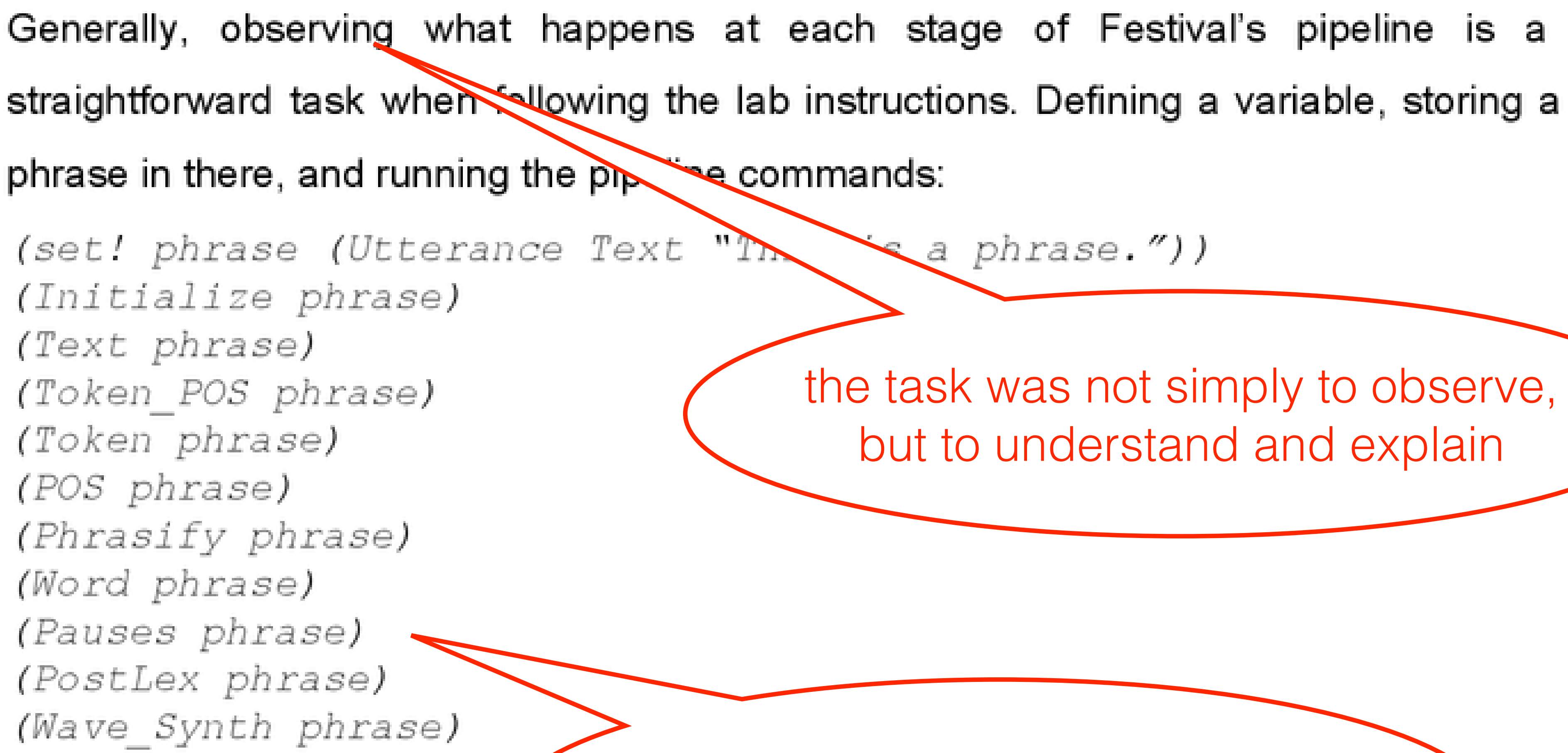
Methodology



The strategy chosen to review Festival was a mixture of an iterative and recursive unit testing: two or three sentences for each class of possible error found (while reviewing the software) were initialised in `Festival` until the sound wave was generated, which was then evaluated with an informal listening test. When an unexpected error was found in terms of intelligibility or naturalness, the initialisation process was restarted, checking the attributes generated at each step, trying to analyse their changes through the utterance generation.

employs a specific methodology,
and uses multiple sentences

Methodology



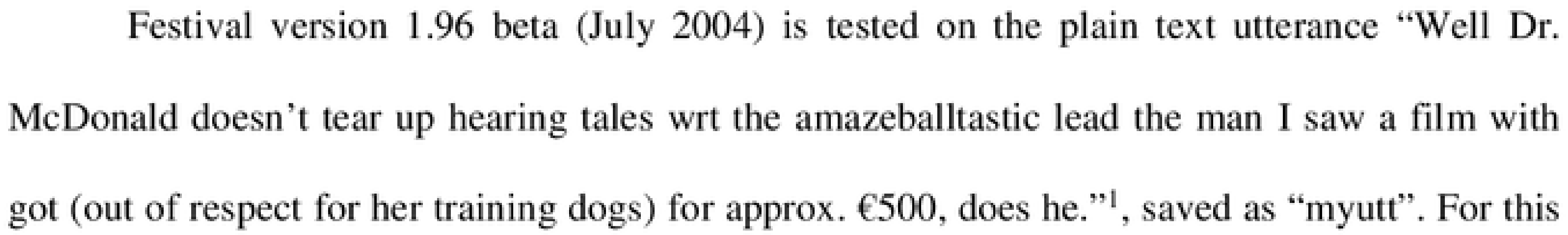
Generally, observing what happens at each stage of Festival's pipeline is a straightforward task when following the lab instructions. Defining a variable, storing a phrase in there, and running the pipeline commands:

```
(set! phrase (Utterance Text "This is a phrase."))
(Initialize phrase)
(Text phrase)
(Token_POS phrase)
(Token phrase)
(POS phrase)
(Phrasify phrase)
(Word phrase)
(Pauses phrase)
(PostLex phrase)
(Wave_Synth phrase)
```

the task was not simply to observe, but to understand and explain

simply repeats something from the lab handout - adds no value to the report

Choice of sentences

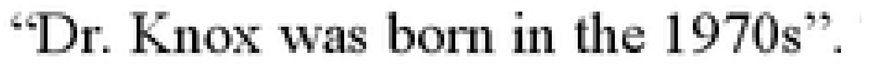


Festival version 1.96 beta (July 2004) is tested on the plain text utterance "Well Dr. McDonald doesn't tear up hearing tales wrt the amazeballtastic lead the man I saw a film with got (out of respect for her training dogs) for approx. €500, does he."[1], saved as "myutt". For this

will cause all modules to do something interesting

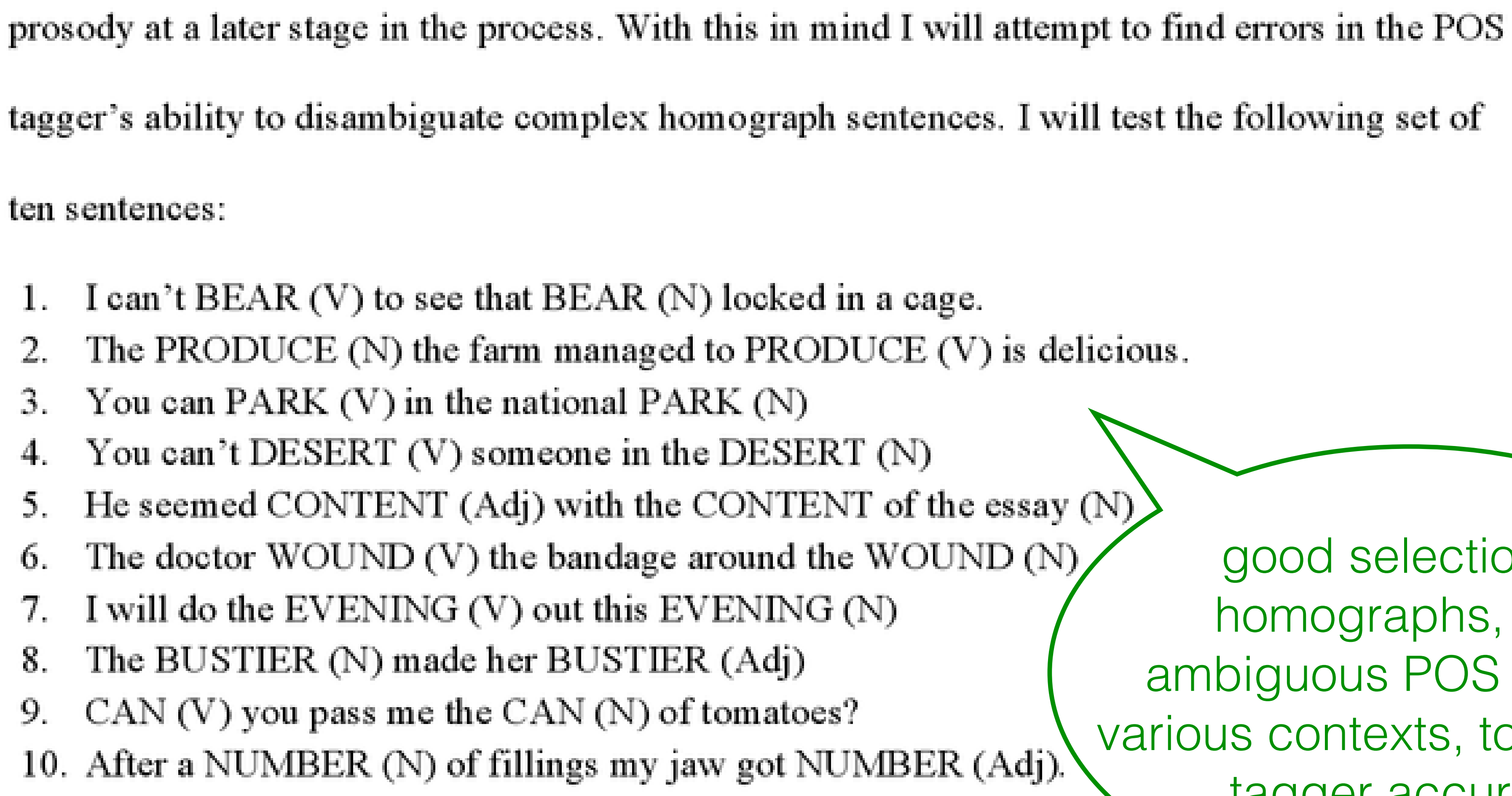
quite long

Choice of sentences


“Dr. Knox was born in the 1970s”.

good choice - requires normalisation of two ambiguous tokens

Choice of sentences

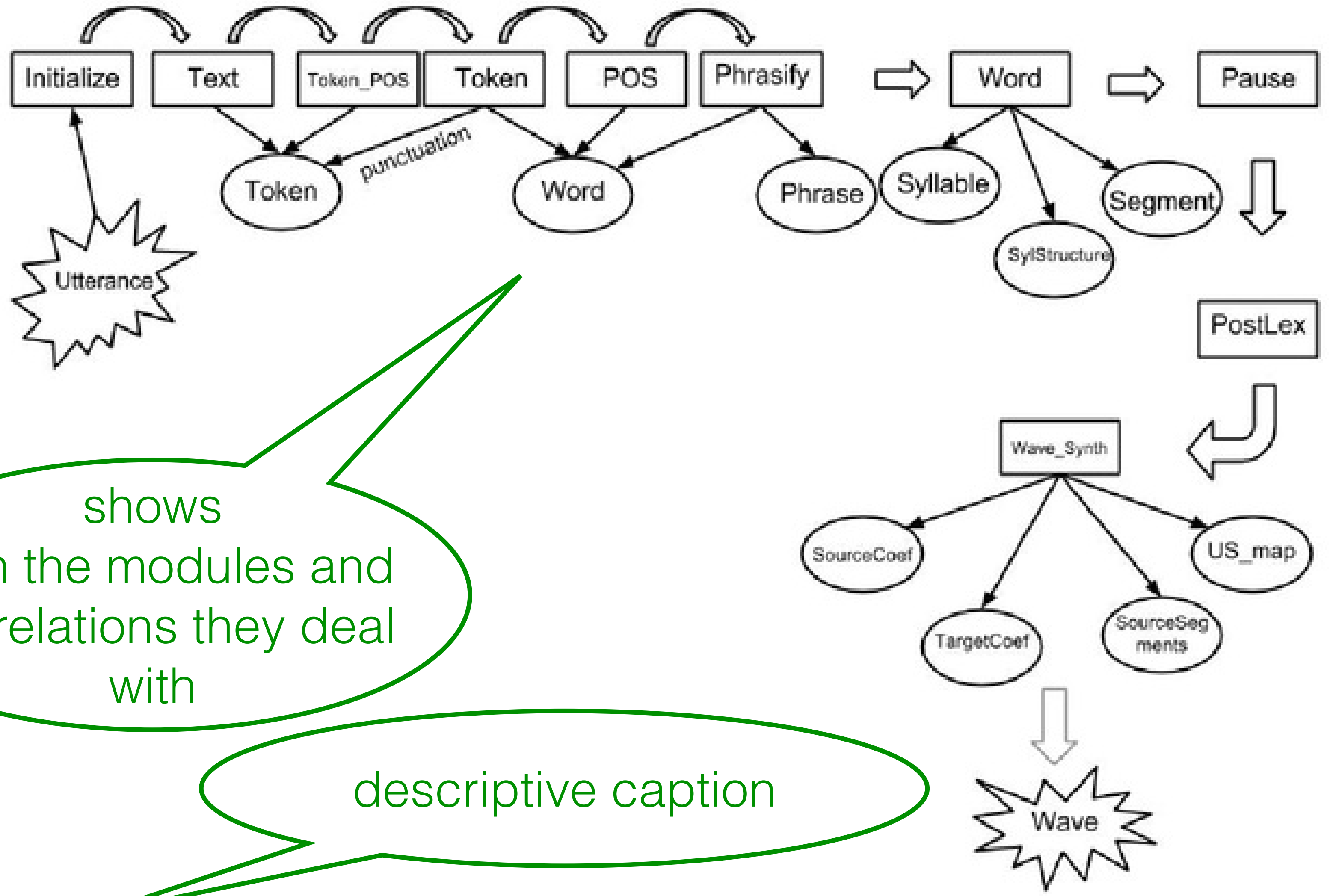


prosody at a later stage in the process. With this in mind I will attempt to find errors in the POS tagger's ability to disambiguate complex homograph sentences. I will test the following set of ten sentences:

1. I can't BEAR (V) to see that BEAR (N) locked in a cage.
2. The PRODUCE (N) the farm managed to PRODUCE (V) is delicious.
3. You can PARK (V) in the national PARK (N)
4. You can't DESERT (V) someone in the DESERT (N)
5. He seemed CONTENT (Adj) with the CONTENT of the essay (N)
6. The doctor WOUND (V) the bandage around the WOUND (N)
7. I will do the EVENING (V) out this EVENING (N)
8. The BUSTIER (N) made her BUSTIER (Adj)
9. CAN (V) you pass me the CAN (N) of tomatoes?
10. After a NUMBER (N) of fillings my jaw got NUMBER (Adj).

good selection of homographs, with ambiguous POS tags, in various contexts, to test POS tagger accuracy

Pipeline



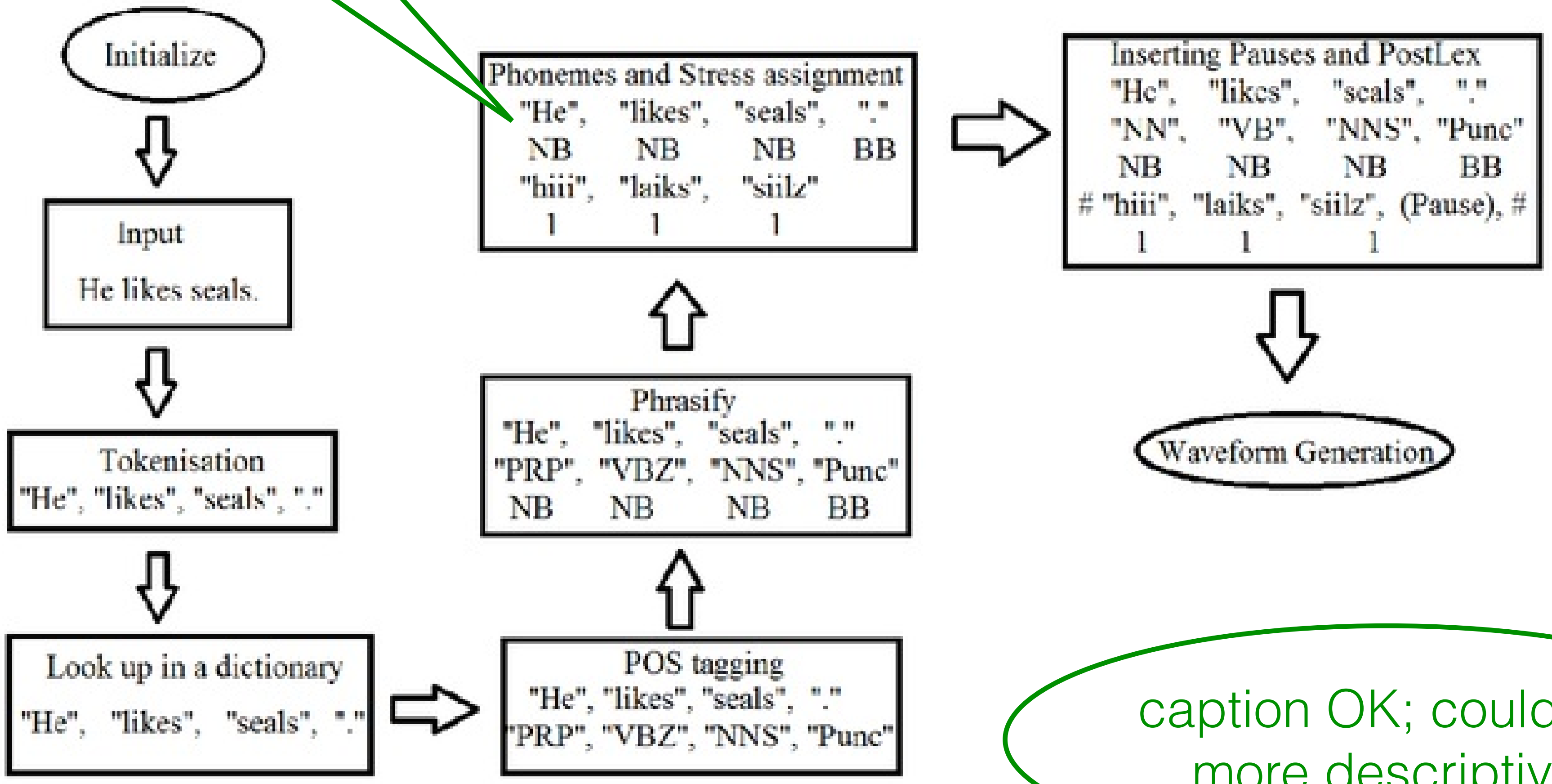
shows both the modules and the relations they deal with

descriptive caption

Figure 1: Schematic workflow of the Festival pipeline. Modules are rectangular, relations oval.

Pipeline

shows a worked example



Initialize
Input
He likes seals.
Tokenisation
"He", "likes", "seals", "."
Look up in a dictionary
"He", "likes", "seals", "."
POS tagging
"He", "likes", "seals", "."
"PRP", "VBZ", "NNS", "Punc"
Phrasify
"He", "likes", "seals", "."
"PRP", "VBZ", "NNS", "Punc"
NB NB NB BB
Phonemes and Stress assignment
"He", "likes", "seals", "."
NB NB NB BB
"hiii", "laiks", "siilz"
1 1 1
Inserting Pauses and PostLex
"He", "likes", "seals", "."
"NN", "VB", "NNS", "Punc"
NB NB NB BB
# "hiii", "laiks", "siilz", (Pause), #
1 1 1
Waveform Generation

caption OK; could be more descriptive

Figure1. Summarisation of speech generation "He likes seals." in Festival

Pipeline

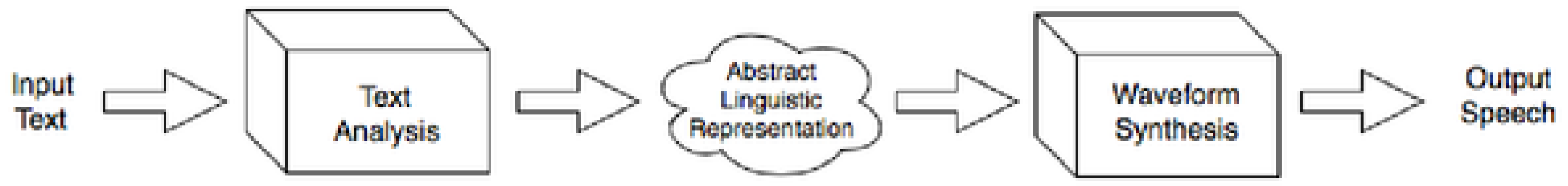
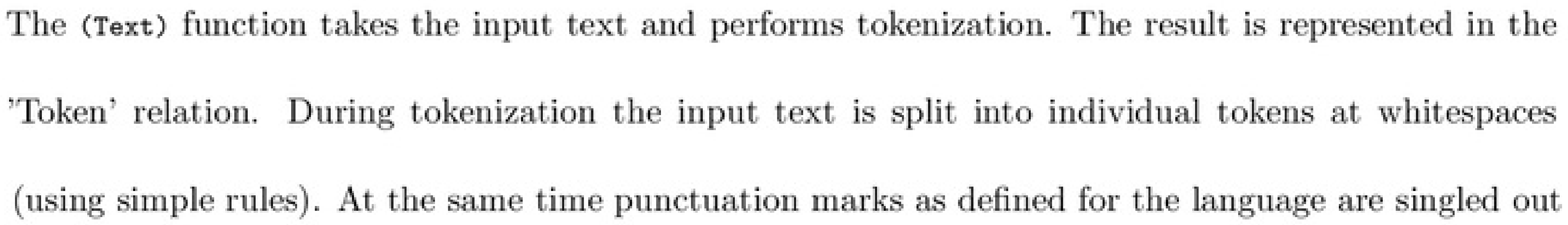


Figure 1.

caption **should** be more descriptive

simple, but helpful

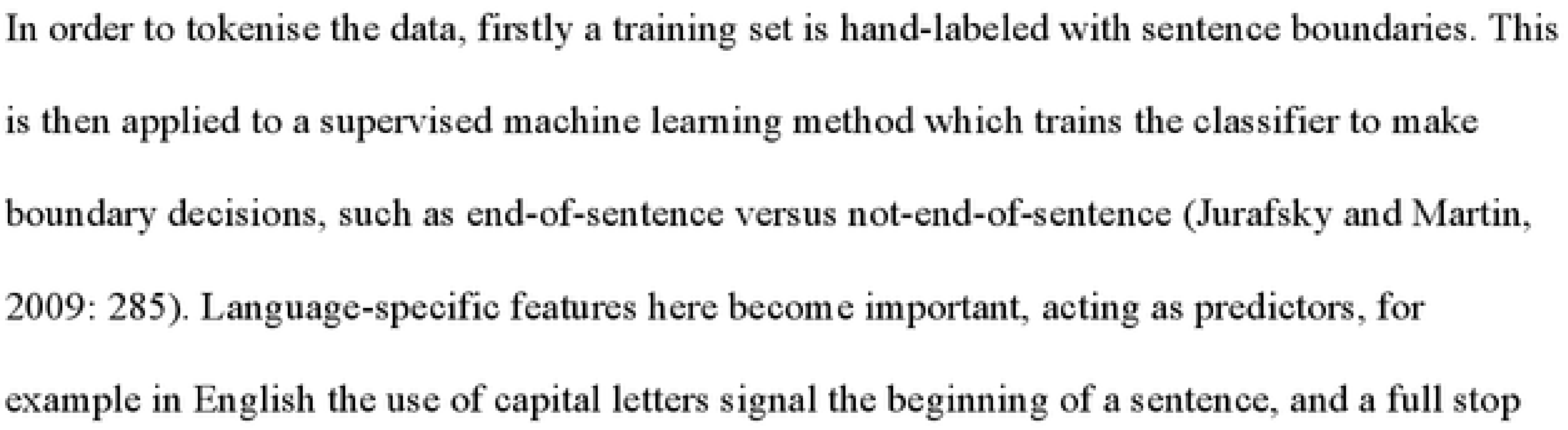
Tokenisation



The (Text) function takes the input text and performs tokenization. The result is represented in the 'Token' relation. During tokenization the input text is split into individual tokens at whitespaces (using simple rules). At the same time punctuation marks as defined for the language are singled out

Says exactly HOW tokenisation is done

Tokenisation

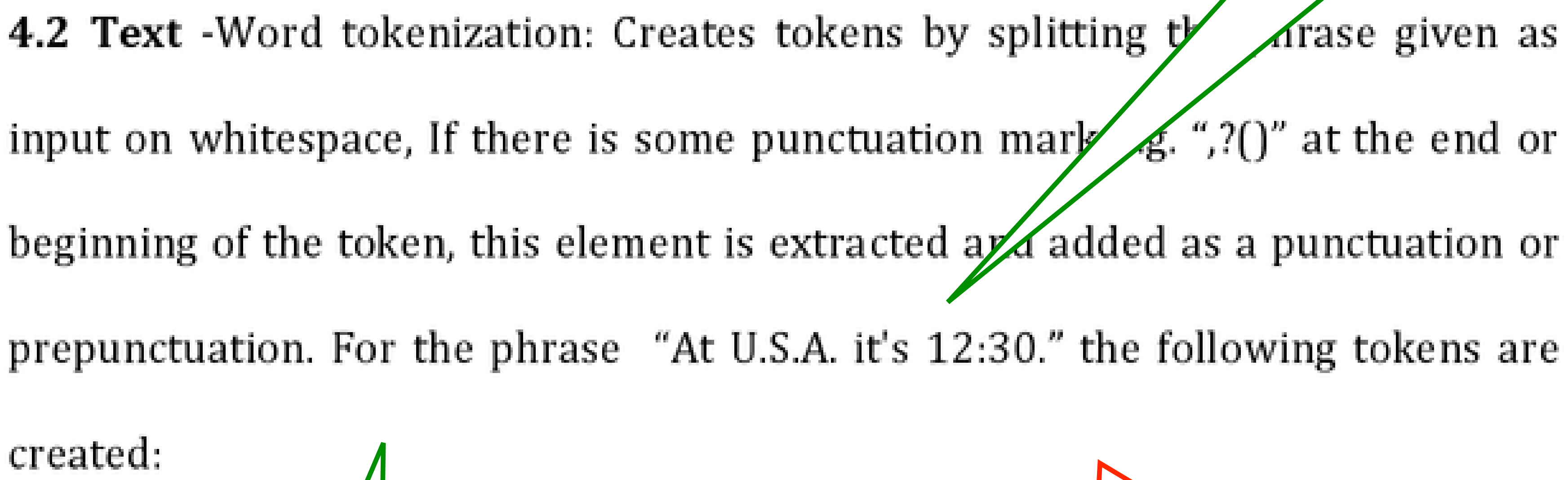


In order to tokenise the data, firstly a training set is hand-labeled with sentence boundaries. This is then applied to a supervised machine learning method which trains the classifier to make boundary decisions, such as end-of-sentence versus not-end-of-sentence (Jurafsky and Martin, 2009: 285). Language-specific features here become important, acting as predictors, for example in English the use of capital letters signal the beginning of a sentence, and a full stop

describes learning from data, including what data is required, how it is labelled, and what predictors might be used

Tokenisation

gives an example

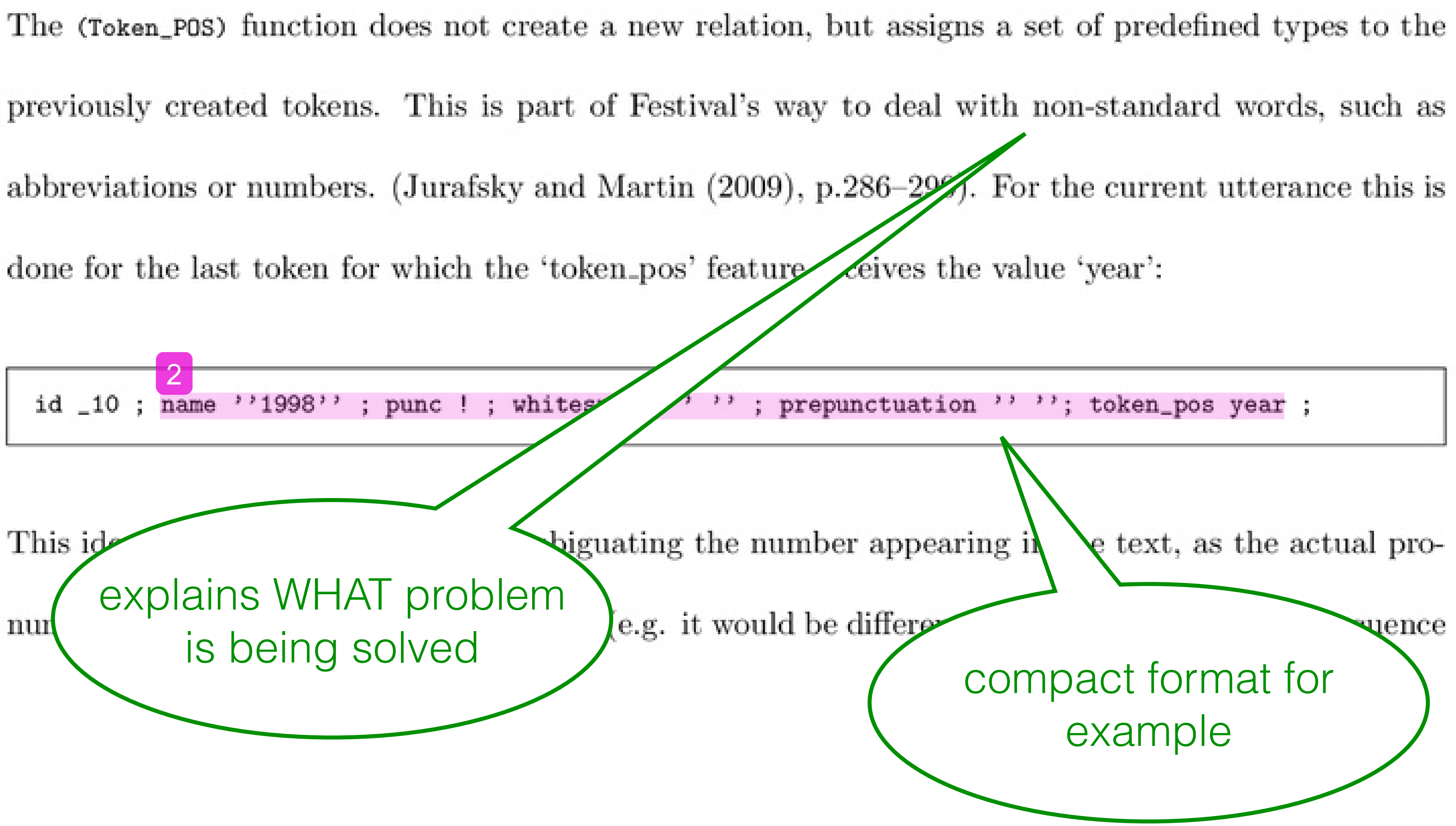


4.2 Text -Word tokenization: Creates tokens by splitting th hrase given as input on whitespace, If there is some punctuation mark g. ",?()" at the end or beginning of the token, this element is extracted ar d added as a punctuation or prepunctuation. For the phrase "At U.S.A. it's 12:30." the following tokens are created:

Says WHAT tokenisation is

should have been explicit about HOW splitting on whitespace is done (is it trivial? complex? rules? a model?)

Non-standard words



The (Token_POS) function does not create a new relation, but assigns a set of predefined types to the previously created tokens. This is part of Festival's way to deal with non-standard words, such as abbreviations or numbers. (Jurafsky and Martin (2009), p.286–29). For the current utterance this is done for the last token for which the 'token_pos' feature ceives the value 'year':

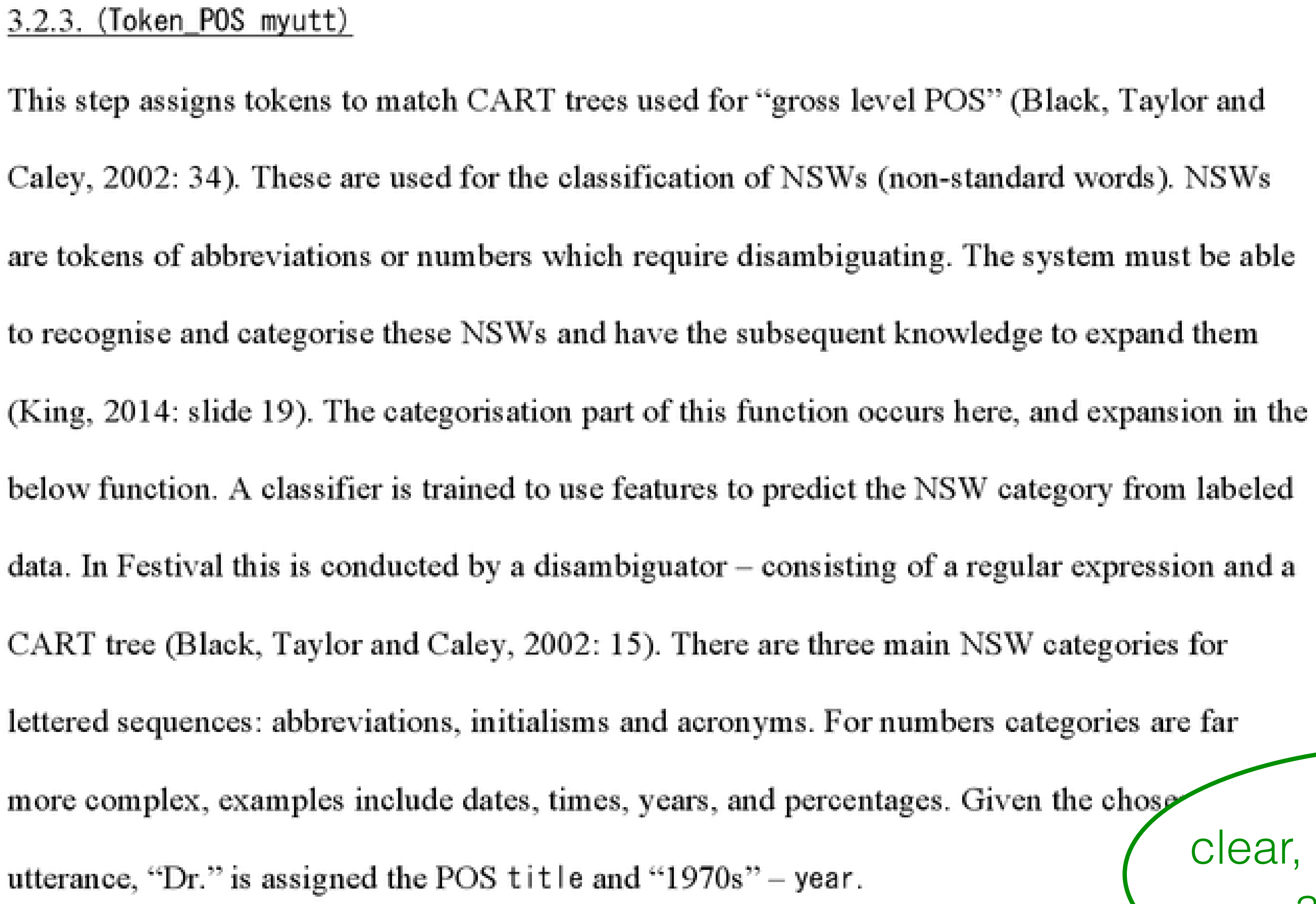
```
id _10 ; name ''1998'' ; punc ! ; whites        '' '' ; prepunctuation '' ''; token_pos year ;
```

This id biguating the number appearing i e text, as the actual pro-

nu (e.g. it would be differe uence

explains WHAT problem is being solved
compact format for example

Non-standard words

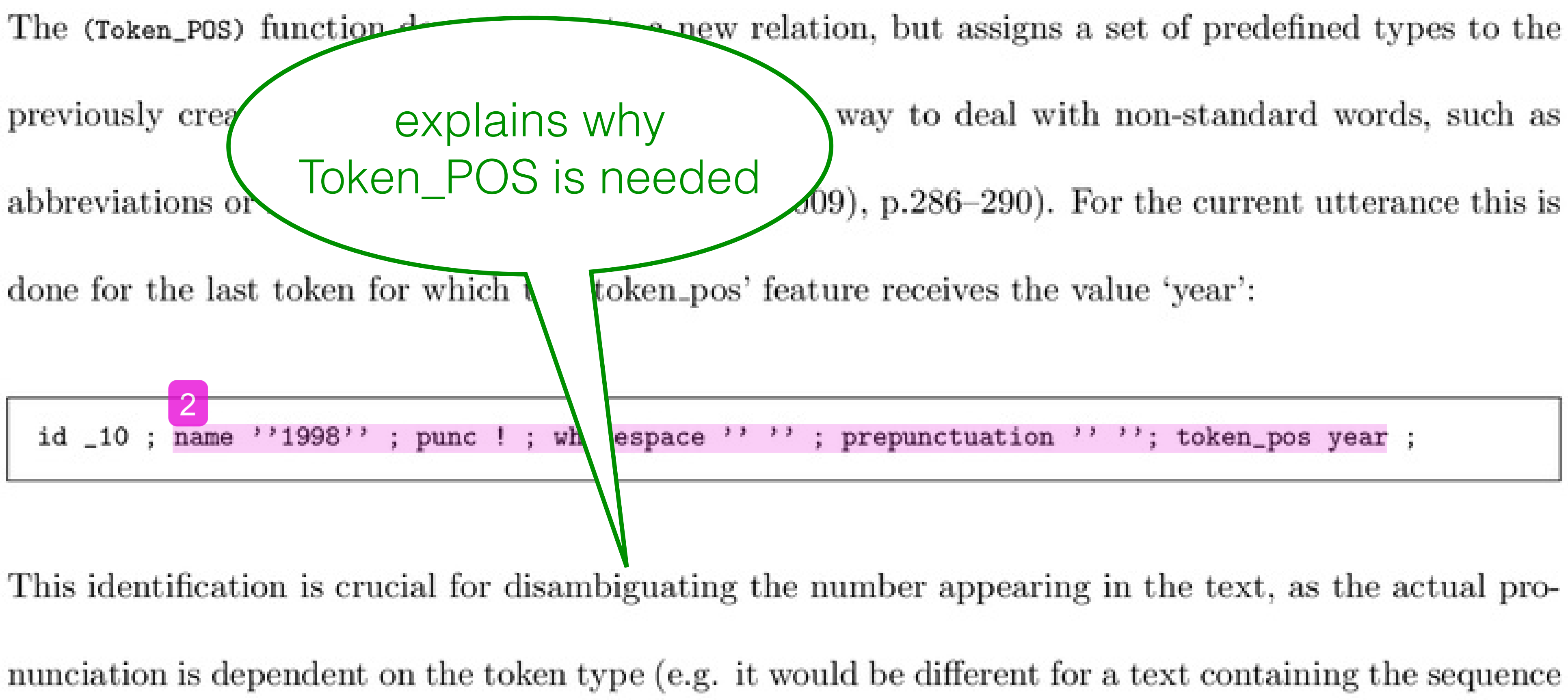


3.2.3. (Token_POS myutt)

This step assigns tokens to match CART trees used for "gross level POS" (Black, Taylor and Caley, 2002: 34). These are used for the classification of NSWs (non-standard words). NSWs are tokens of abbreviations or numbers which require disambiguating. The system must be able to recognise and categorise these NSWs and have the subsequent knowledge to expand them (King, 2014: slide 19). The categorisation part of this function occurs here, and expansion in the below function. A classifier is trained to use features to predict the NSW category from labeled data. In Festival this is conducted by a disambiguator – consisting of a regular expression and a CART tree (Black, Taylor and Caley, 2002: 15). There are three main NSW categories for lettered sequences: abbreviations, initialisms and acronyms. For numbers categories are far more complex, examples include dates, times, years, and percentages. Given the chose utterance, "Dr." is assigned the POS title and "1970s" – year.

clear, concise, precise and detailed

Non-standard words



The (Token_POS) function d a new relation, but assigns a set of predefined types to the previously crea way to deal with non-standard words, such as abbreviations or 09), p.286–290). For the current utterance this is done for the last token for which t token_pos' feature receives the value 'year':

explains why Token_POS is needed

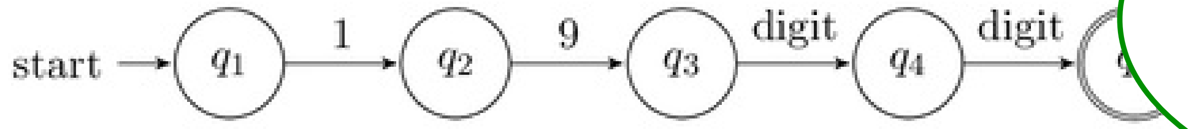
2

```
id _10 ; name ''1998'' ; punc ! ; wh espace '' '' ; prepunctuation '' ''; token_pos year ;
```

This identification is crucial for disambiguating the number appearing in the text, as the actual pronunciation is dependent on the token type (e.g. it would be different for a text containing the sequence

Non-standard words

1998 Euros, i.e. it is context-sensitive). The type identification thus has to proceed in two steps. The token to disambiguate first has to be recognised. This can be achieved using regular expressions (Regex) or equivalently finite state machines (FSM) tha[illegible] match to or accept certain strings and reject others. (King (2014), slides 19–21) An example FSM that reco[illegible]s strings denoting a year between 1900 and 1999 is shown below.



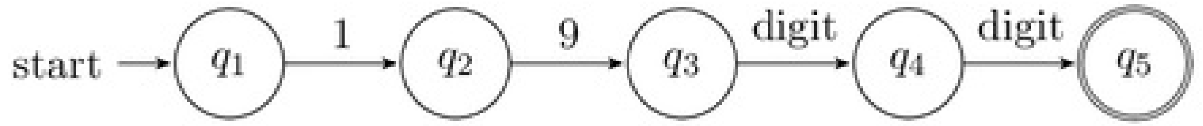
Explains that there are two steps: detect, then classify into types

For tokens that match a regular expression a Classification and Regression Tree (CART) is applied to assign a type class to that token.[2] In Festival these Regex-CART-pairs are called disambiguators.

ndard words

then separately explains each step: detect using regex, then classify using CART

1998 e). The type identification thus has to proceed in two steps. The token to disambiguate first has to be ognised. This can be achieved using regular expressions (Regex) or equivalently finite state machines (FSM) that match to or accept certain strings and reject others. (King (2014), slides 19–21) An example FSM that recognises strings denoting a year between 1900 and 1999 is shown below.



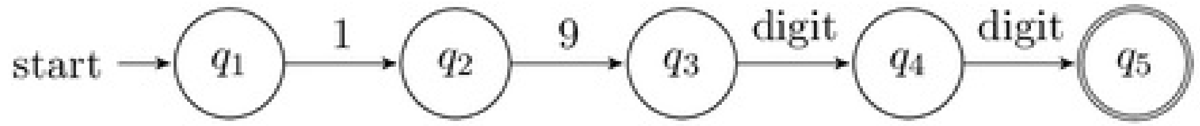
$$\begin{tikzcd}
\text{start} \arrow[r] & q_1 \arrow[r, "1"] & q_2 \arrow[r, "9"] & q_3 \arrow[r, "\text{digit}"] & q_4 \arrow[r, "\text{digit}"] & q_5
\end{tikzcd}$$

For tokens that match a regular expression a Classification and Regression Tree (CART) is applied to assign a type class to that token.[2] In Festival these Regex-CART-pairs are called disambiguators.

Non-standard words

1998 Euros, i.e. it is context-sensitive). The type identification thus h . The
token to disambiguate first has to be recognised. This can
(Regex) or equivalently finite state machines (FSM) that mat
others. (King (2014), slides 19–21) An example FSM that recogn n
1900 and 1999 is shown below.

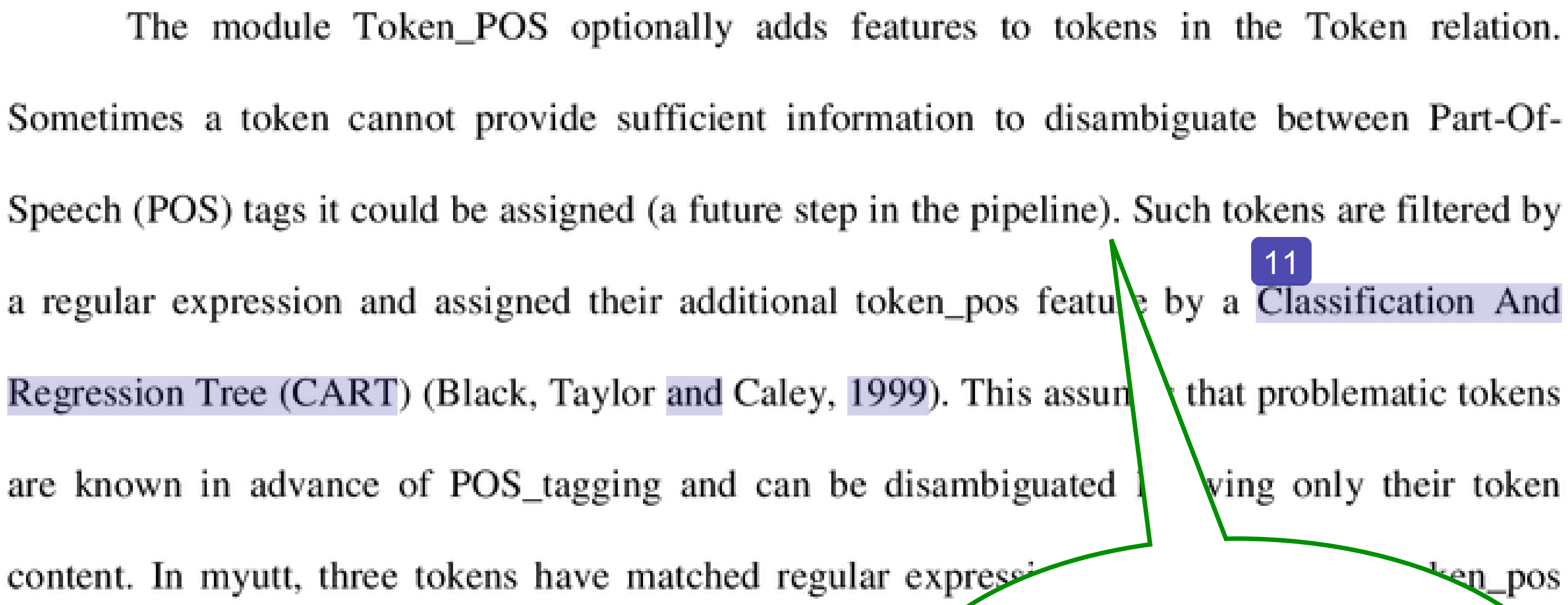
compact example of a regex to detect years, expressed as a FSM



$$\begin{tikzcd}\text{start} \arrow[r] & q_1 \arrow[r, "1"] & q_2 \arrow[r, "9"] & q_3 \arrow[r, "\text{digit}"] & q_4 \arrow[r, "\text{digit}"] & \boxed{q_5}\end{tikzcd}$$

For tokens that match a regular expression a Classification and Regression Tree (CART) is applied to assign a type class to that token.[2] In Festival these Regex-CART-pairs are called disambiguators.

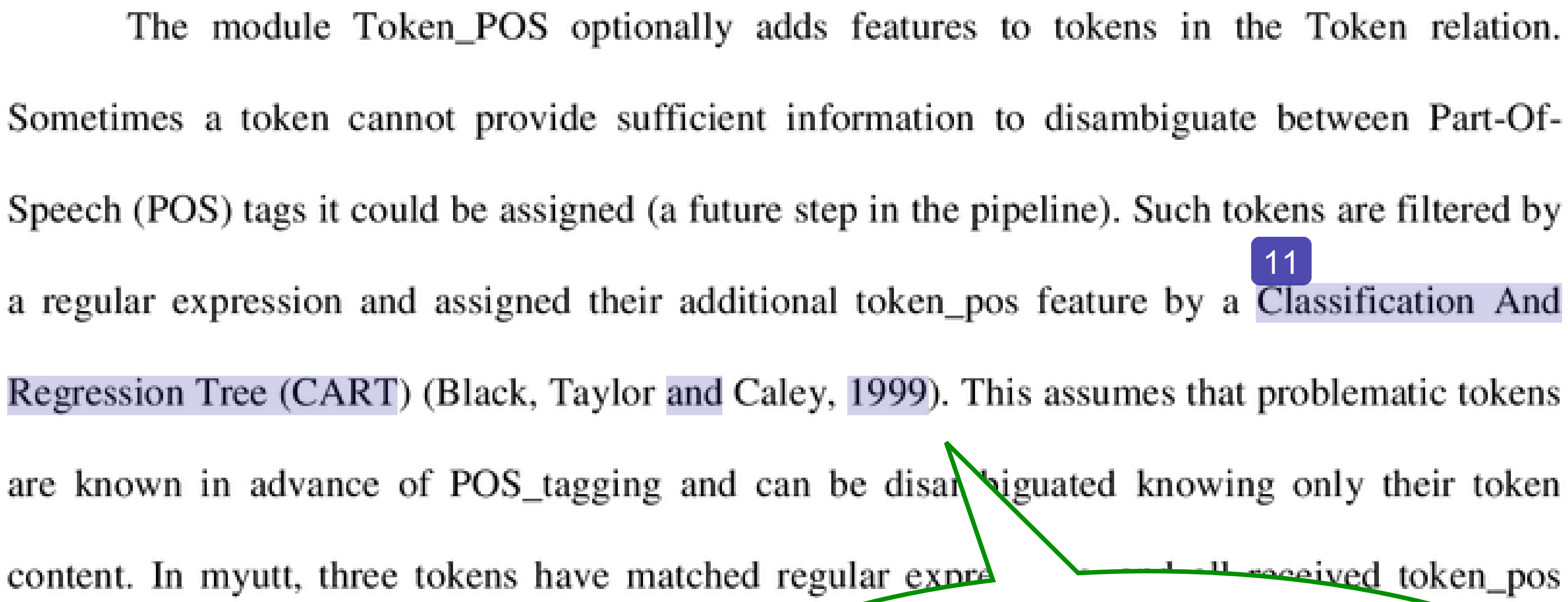
Non-standard words



The module Token_POS optionally adds features to tokens in the Token relation. Sometimes a token cannot provide sufficient information to disambiguate between Part-Of-Speech (POS) tags it could be assigned (a future step in the pipeline). Such tokens are filtered by a regular expression and assigned their additional token_pos featu e by a 11 Classification And Regression Tree (CART) (Black, Taylor and Caley, 1999). This assun that problematic tokens are known in advance of POS_tagging and can be disambiguated l ving only their token content. In myutt, three tokens have matched regular express ken_pos

understands limitations of conventional POS tagging

Non-standard words

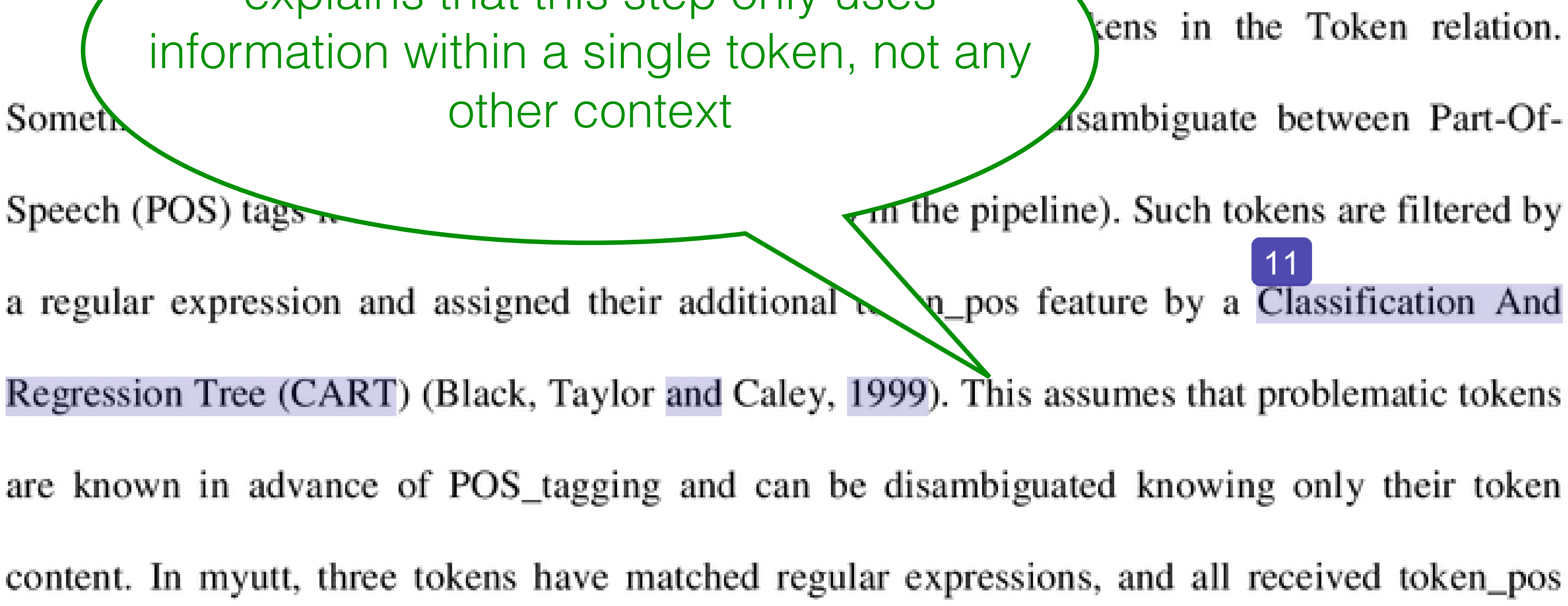


The module Token_POS optionally adds features to tokens in the Token relation. Sometimes a token cannot provide sufficient information to disambiguate between Part-Of-Speech (POS) tags it could be assigned (a future step in the pipeline). Such tokens are filtered by a regular expression and assigned their additional token_pos feature by a Classification And Regression Tree (CART) (Black, Taylor and Caley, 1999). This assumes that problematic tokens are known in advance of POS_tagging and can be disar biguated knowing only their token content. In myutt, three tokens have matched regular expre received token_pos

another nice explanation of filtering (detection) using regex, then classification (assigning token_pos) using CART

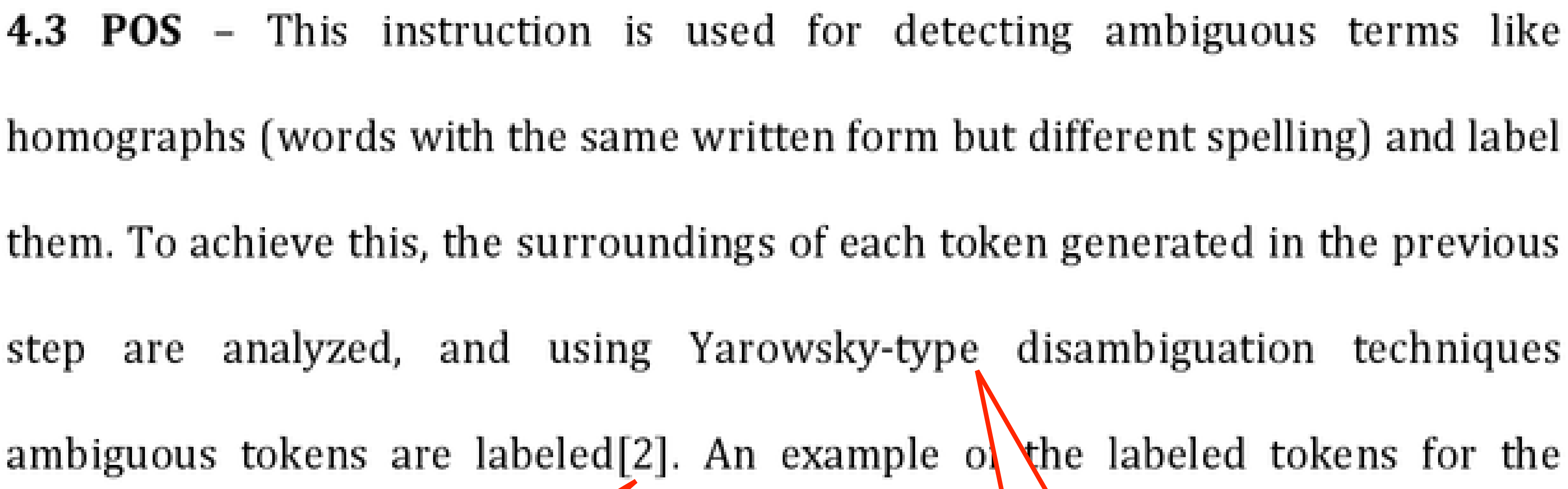
Non-standard words

explains that this step only uses information within a single token, not any other context



kens in the Token relation. Someti isambiguate between Part-Of-Speech (POS) tags in the pipeline). Such tokens are filtered by a regular expression and assigned their additional n_pos feature by a 11 Classification And Regression Tree (CART) (Black, Taylor and Caley, 1999). This assumes that problematic tokens are known in advance of POS_tagging and can be disambiguated knowing only their token content. In myutt, three tokens have matched regular expressions, and all received token_pos

Non-standard words

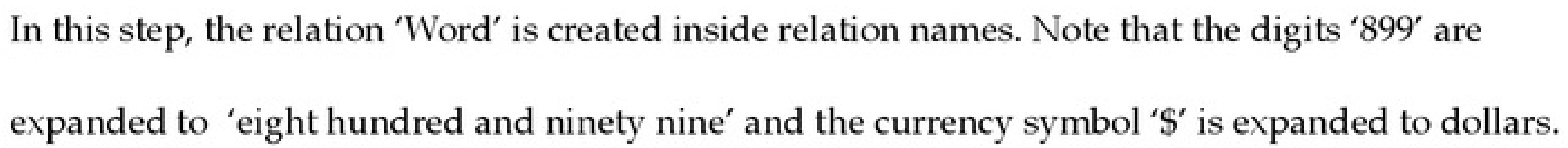


4.3 POS – This instruction is used for detecting ambiguous terms like homographs (words with the same written form but different spelling) and label them. To achieve this, the surroundings of each token generated in the previous step are analyzed, and using Yarowsky-type disambiguation techniques ambiguous tokens are labeled[2]. An example of the labeled tokens for the

cites Festival manual without a specific section

not clear the writer knows what "Yarowsky-type disambiguation techniques" are - they are not explained here

Non-standard words

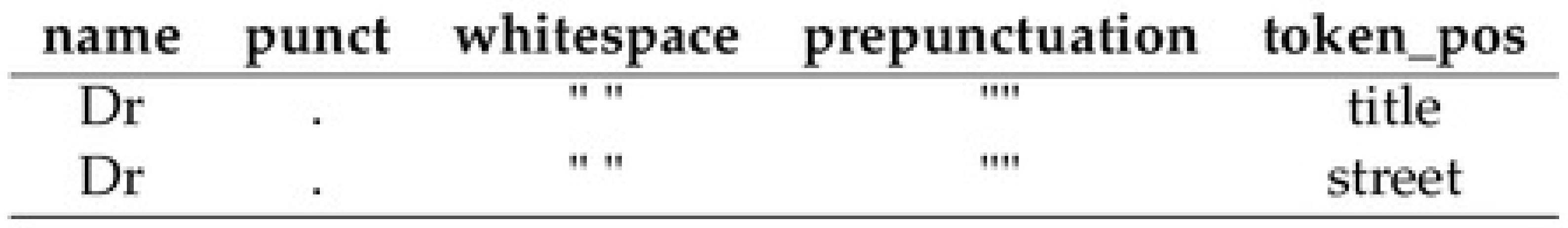


In this step, the relation 'Word' is created inside relation names. Note that the digits '899' are expanded to 'eight hundred and ninety nine' and the currency symbol '$' is expanded to dollars.

no details on **how** any of this is achieved

no demonstration that the writer **understands** how it is done, or what the **limitations** of the method are

Non-standard words



name	punct	whitespace	prepunctuation	token_pos
Dr	.	" "	""	title
Dr	.	" "	""	street

Table 2: Disambiguation of Dr. in Token relation

good attempt to use a table to summarise findings

but **no context given**, so we can't tell why one is a street and the other a title

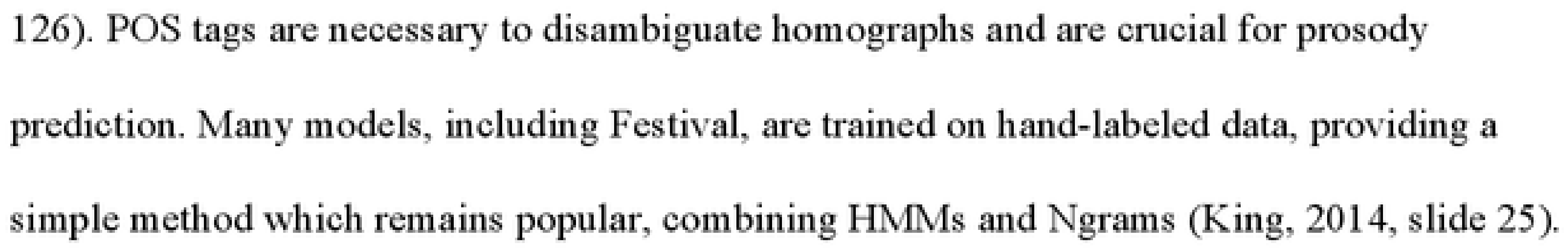
POS tagging

Festival utilises a (probabilistic) Hidden-Markov-Model POS Tagger (Black et al. (1999), section 16) which makes use of a simplifying assumption about sequence probabilities by assigning tags to words based on tag transition and word output probabilities. (Jurafsky and Martin (2009), chapter 5.5) For the current sentence there is a wrong assignment of e tag 'jj' for adjectives (as opposed to 'cc' for cardinal numbers) to the first number in the expanded ye

```
id _21 ; name nineteen ; pos_index 5 ; pos_index_score 0 ; pos jj

id _22 ; name ninety ; pos_index 7 ; pos_index_score 0 ; pos cd

id _23 ; name eight ; pos_index 7 ; pos_index_score 0 ; pos
```

describes method used and its limitations

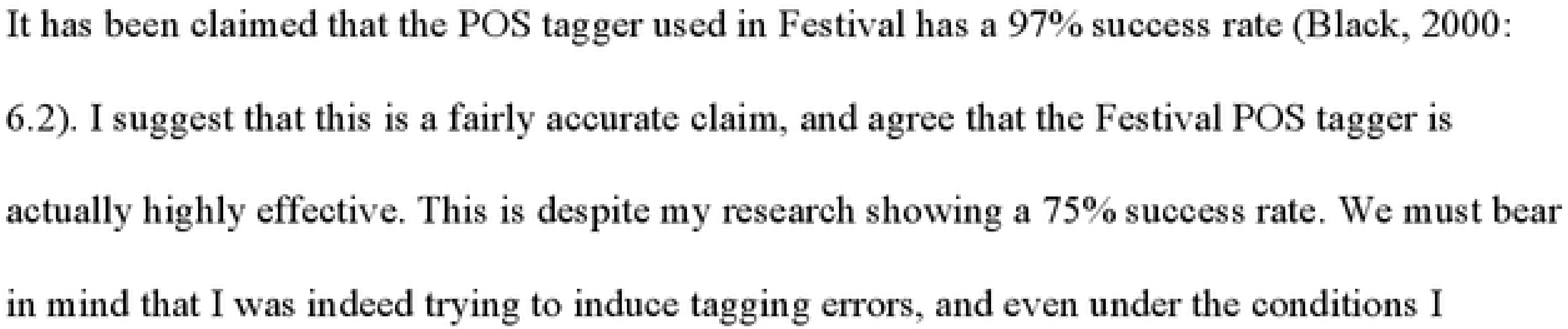
POS tagging



126). POS tags are necessary to disambiguate homographs and are crucial for prosody prediction. Many models, including Festival, are trained on hand-labeled data, providing a simple method which remains popular, combining HMMs and Ngrams (King, 2014, slide 25).

explains why POS tags are needed and what data you need in order to train a tagger

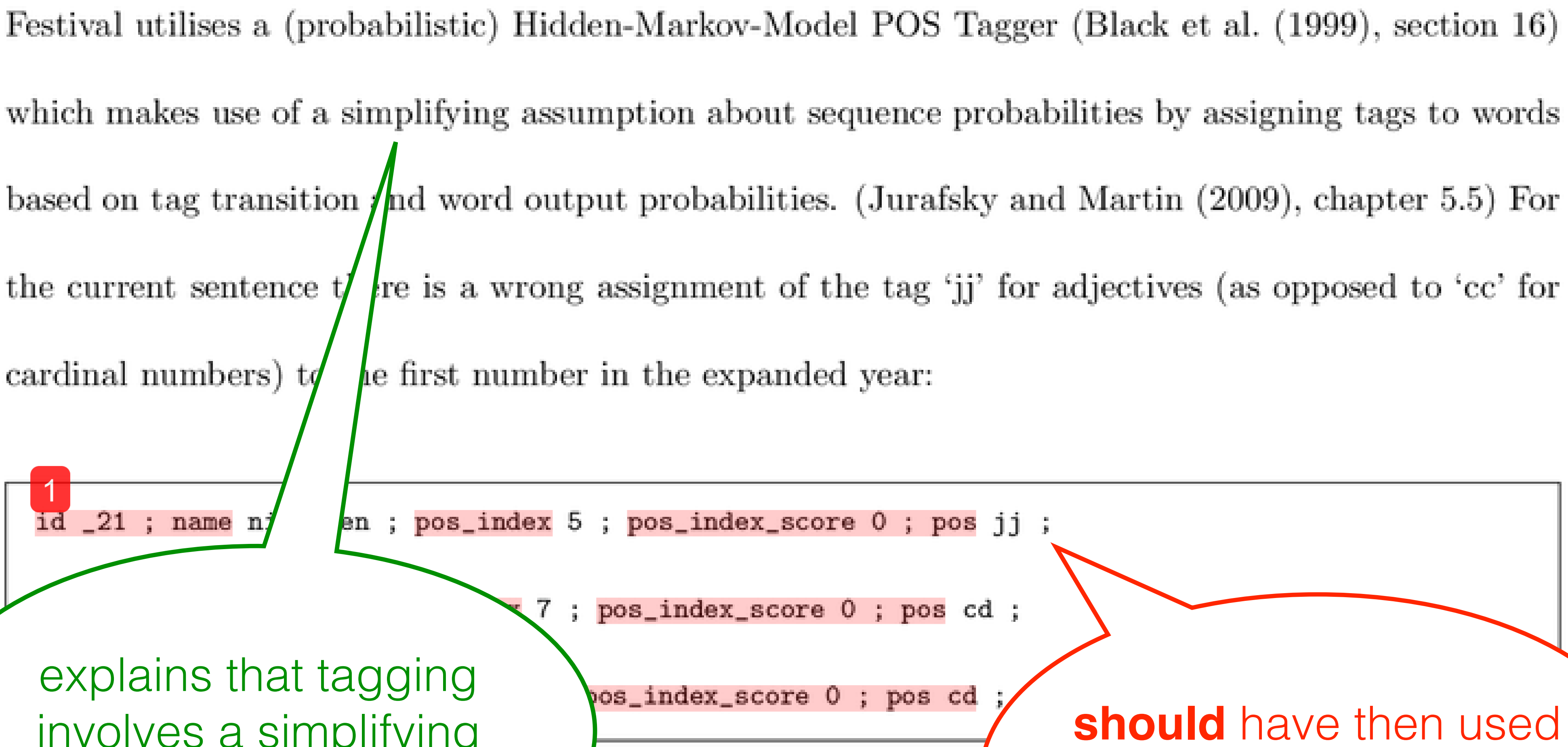
POS tagging



It has been claimed that the POS tagger used in Festival has a 97% success rate (Black, 2000: 6.2). I suggest that this is a fairly accurate claim, and agree that the Festival POS tagger is actually highly effective. This is despite my research showing a 75% success rate. We must bear in mind that I was indeed trying to induce tagging errors, and even under the conditions I

good methodology and clear reporting of results, plus a comparison to a published figure

POS tagging



Festival utilises a (probabilistic) Hidden-Markov-Model POS Tagger (Black et al. (1999), section 16) which makes use of a simplifying assumption about sequence probabilities by assigning tags to words based on tag transition and word output probabilities. (Jurafsky and Martin (2009), chapter 5.5) For the current sentence there is a wrong assignment of the tag 'jj' for adjectives (as opposed to 'cc' for cardinal numbers) to the first number in the expanded year:

1

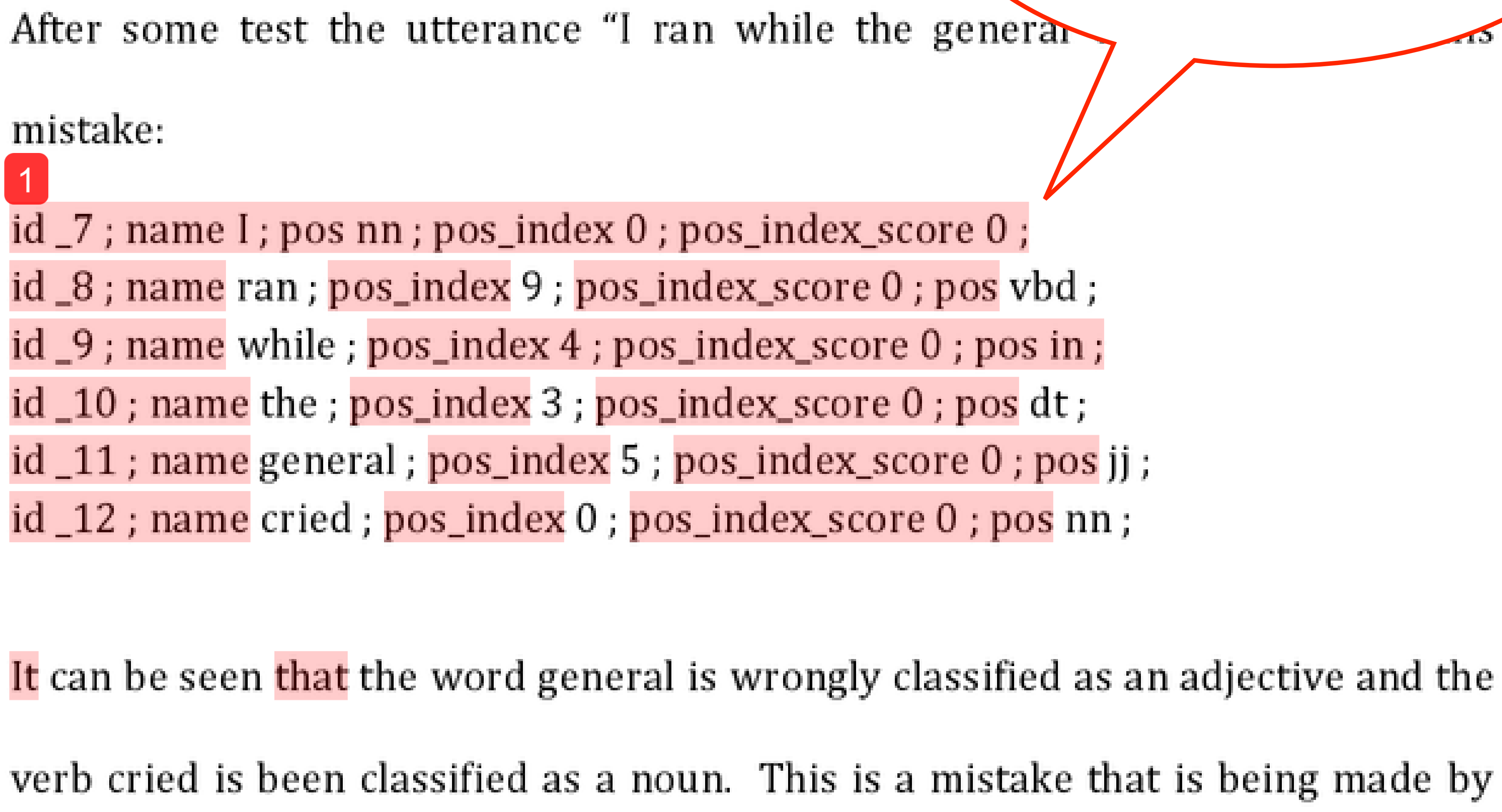
```
id _21 ; name ni    en ; pos_index 5 ; pos_index_score 0 ; pos jj ;

                         7 ; pos_index_score 0 ; pos cd ;

                         os_index_score 0 ; pos cd ;
```

explains that tagging involves a simplifying assumption

should have then used that to try to explain why this error occurred

POS tagg

very wasteful way of showing the POS tags - lots of unnecessary information



After some test the utterance “I ran while the general

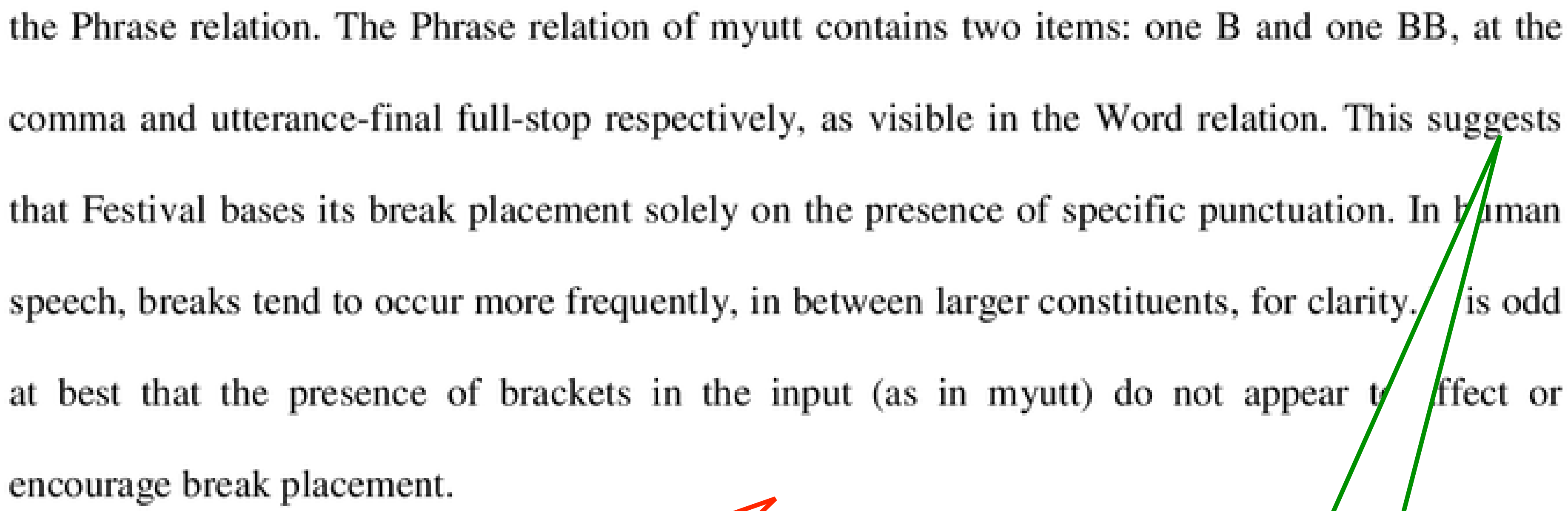
mistake:

1

id _7 ; name I ; pos nn ; pos_index 0 ; pos_index_score 0 ;
id _8 ; name ran ; pos_index 9 ; pos_index_score 0 ; pos vbd ;
id _9 ; name while ; pos_index 4 ; pos_index_score 0 ; pos in ;
id _10 ; name the ; pos_index 3 ; pos_index_score 0 ; pos dt ;
id _11 ; name general ; pos_index 5 ; pos_index_score 0 ; pos jj ;
id _12 ; name cried ; pos_index 0 ; pos_index_score 0 ; pos nn ;

It can be seen that the word general is wrongly classified as an adjective and the verb cried is been classified as a noun. This is a mistake that is being made by

Phrasing

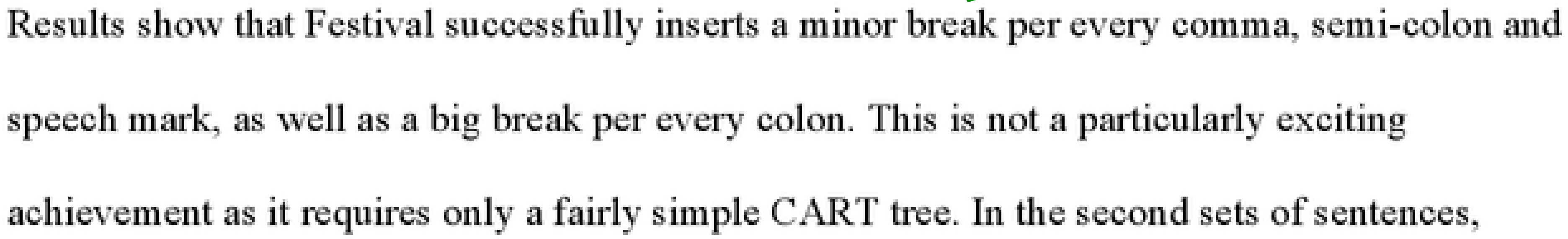

the Phrase relation. The Phrase relation of myutt contains two items: one B and one BB, at the comma and utterance-final full-stop respectively, as visible in the Word relation. This suggests that Festival bases its break placement solely on the presence of specific punctuation. In human speech, breaks tend to occur more frequently, in between larger constituents, for clarity. It is odd at best that the presence of brackets in the input (as in myutt) do not appear to affect or encourage break placement.

could have provided further examples to illustrate this

fair criticism of Festival's reliance on punctuation

Phrasing

correct analysis of break types at certain punctuation marks

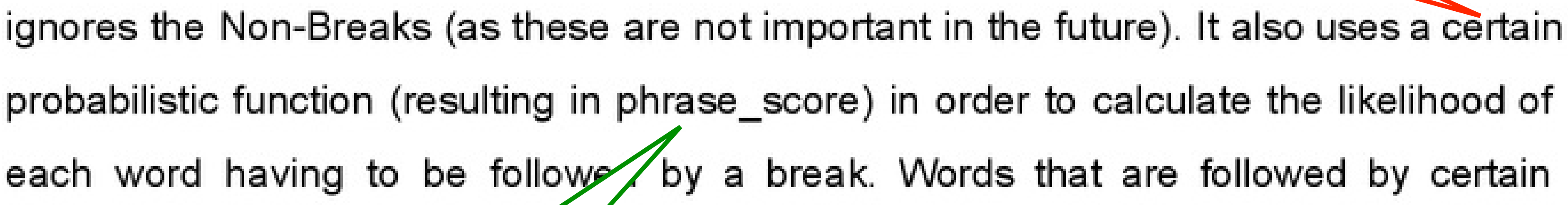


Results show that Festival successfully inserts a minor break per every comma, semi-colon and speech mark, as well as a big break per every colon. This is not a particularly exciting achievement as it requires only a fairly simple CART tree. In the second sets of sentences,

could have drawn this “fairly simple CART tree”

Phrasing

but “certain probabilistic function” is too vague - what form could it take? a CART? something else?



ignores the Non-Breaks (as these are not important in the future). It also uses a certain probabilistic function (resulting in phrase_score) in order to calculate the likelihood of each word having to be followed by a break. Words that are followed by certain

correctly identifies what phrase_score means

Phrasing

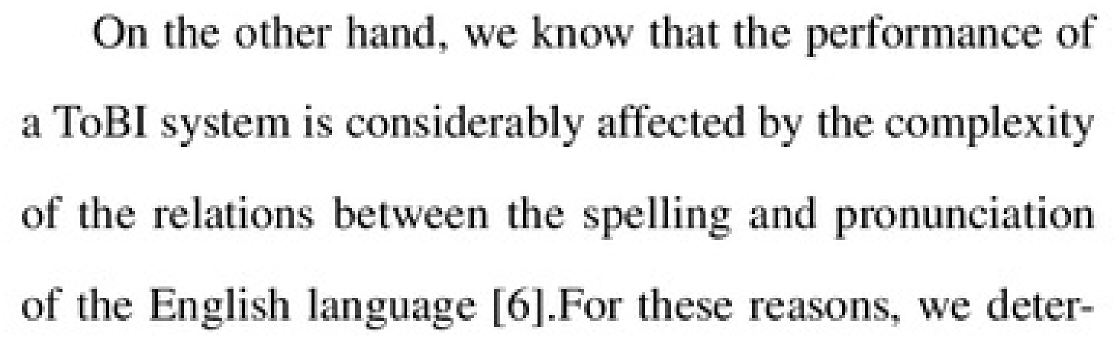
()

id _15 ; name BB ;

nil

meaningless example: not annotated, not clear what the point is since we don’t know how this aligns with the words

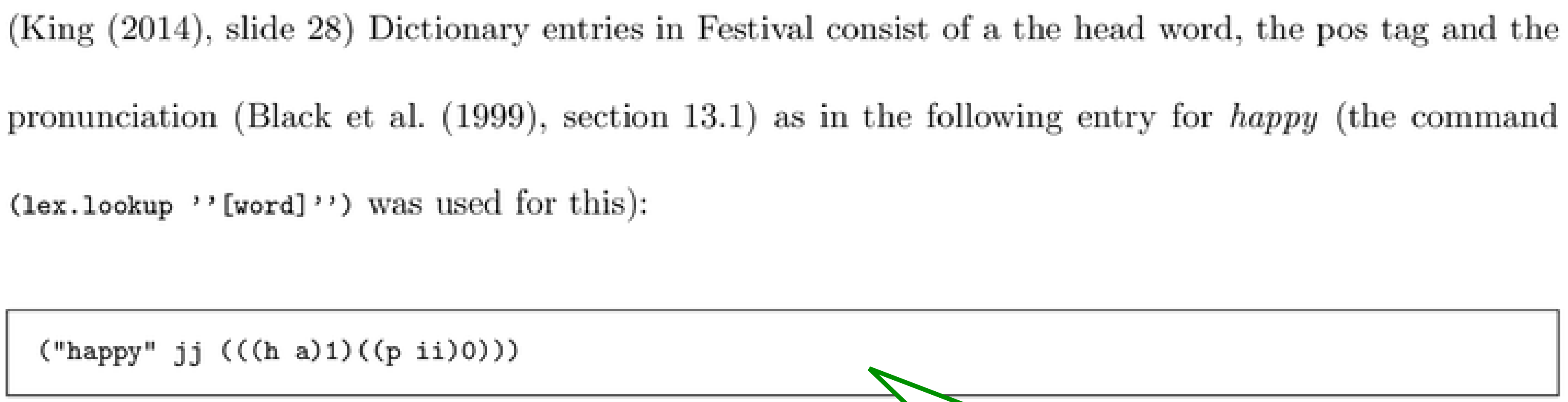
Phrasing



On the other hand, we know that the performance of a ToBI system is considerably affected by the complexity of the relations between the spelling and pronunciation of the English language [6].For these reasons, we deter-

ToBI is nothing to do with letter-to-sound relationships

Lexicon, LTS

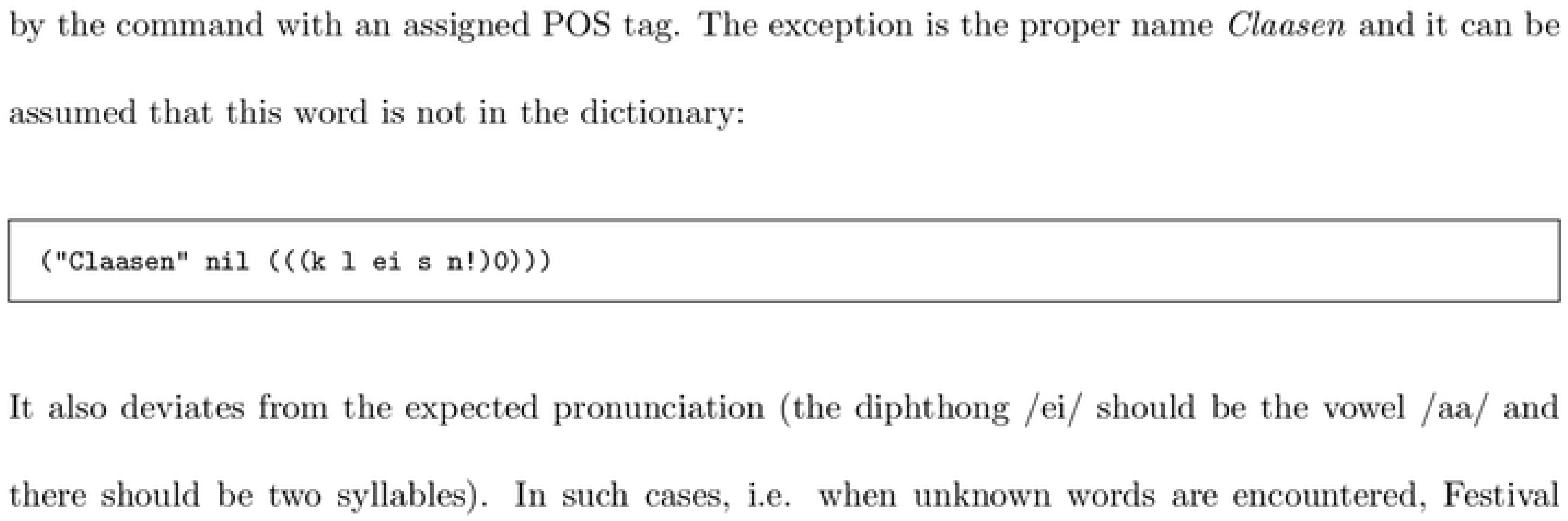


(King (2014), slide 28) Dictionary entries in Festival consist of a the head word, the pos tag and the pronunciation (Black et al. (1999), section 13.1) as in the following entry for *happy* (the command `(lex.lookup ''[word]'')` was used for this):

```
("happy" jj (((h a)1)((p ii)0)))
```

compact description of what lexical entries contain

Lexicon, LTS



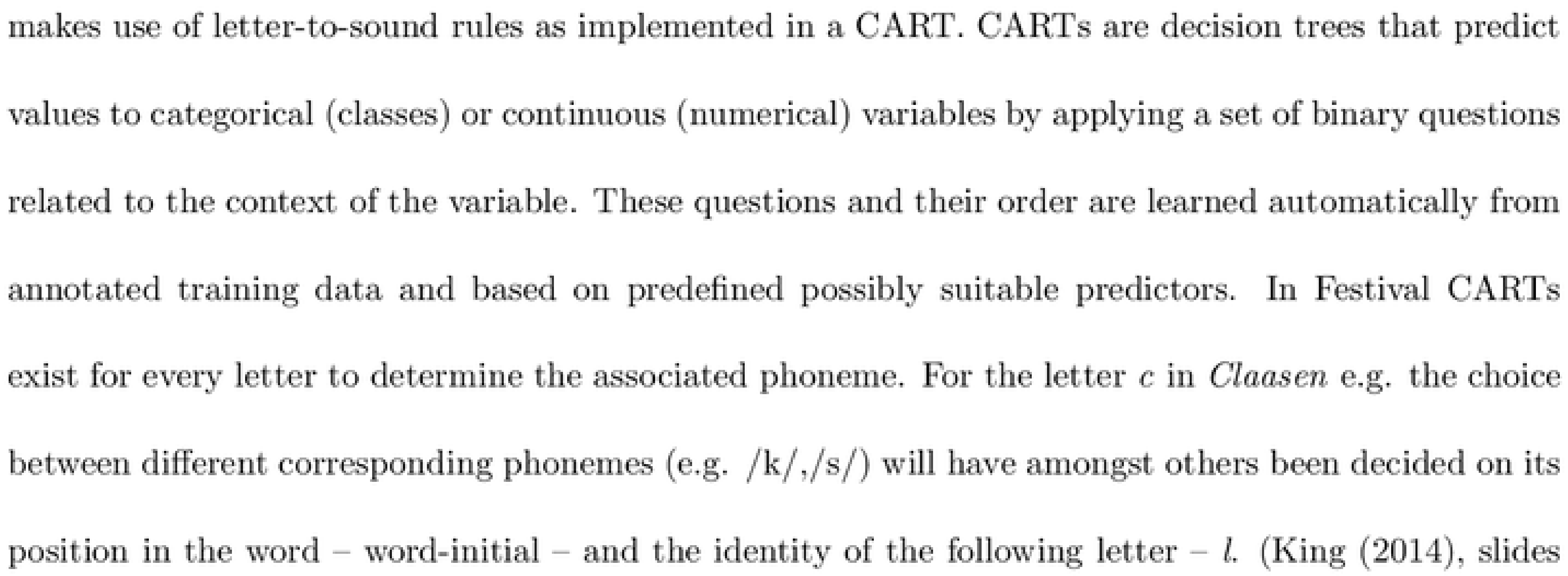
by the command with an assigned POS tag. The exception is the proper name *Claasen* and it can be assumed that this word is not in the dictionary:

```
("Claasen" nil (((k l ei s n!)0)))
```

It also deviates from the expected pronunciation (the diphthong /ei/ should be the vowel /aa/ and there should be two syllables). In such cases, i.e. when unknown words are encountered, Festival

example of out-of-vocabulary word, including Festival's prediction and explanation of what correct pronunciation should have been

Lexicon, LTS

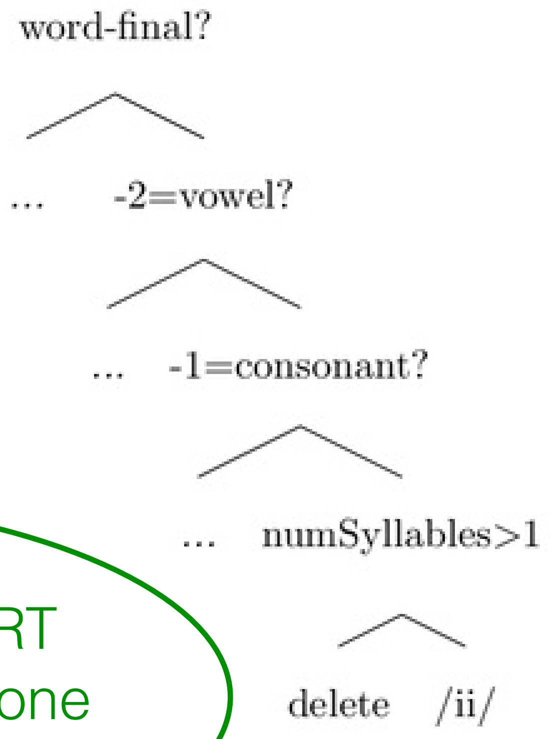


makes use of letter-to-sound rules as implemented in a CART. CARTs are decision trees that predict values to categorical (classes) or continuous (numerical) variables by applying a set of binary questions related to the context of the variable. These questions and their order are learned automatically from annotated training data and based on predefined possibly suitable predictors. In Festival CARTs exist for every letter to determine the associated phoneme. For the letter *c* in *Claasen* e.g. the choice between different corresponding phonemes (e.g. /k/,/s/) will have amongst others been decided on its position in the word – word-initial – and the identity of the following letter – *l*. (King (2014), slides

description of how CARTs are used for LTS, what the predictees and predictors are

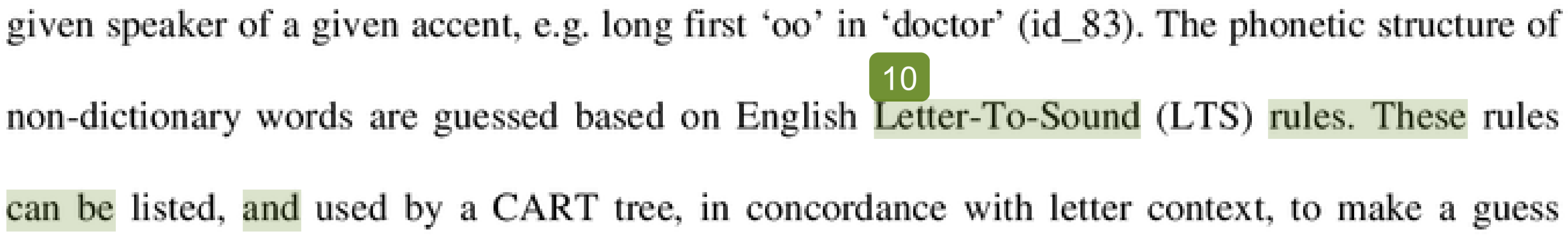
Lexicon, LTS

pronunciation of word-final *e*. Such rules could possibly include syllable length, as *syncope* differs in this respect from standard English words with silent *e*. An example for such a CART containing rules for the letter *e* is shown below (right branches denote yes-anwers; left branches denote no-answers).



example of how the CART could be improved to fix one particular error

Lexicon, LTS



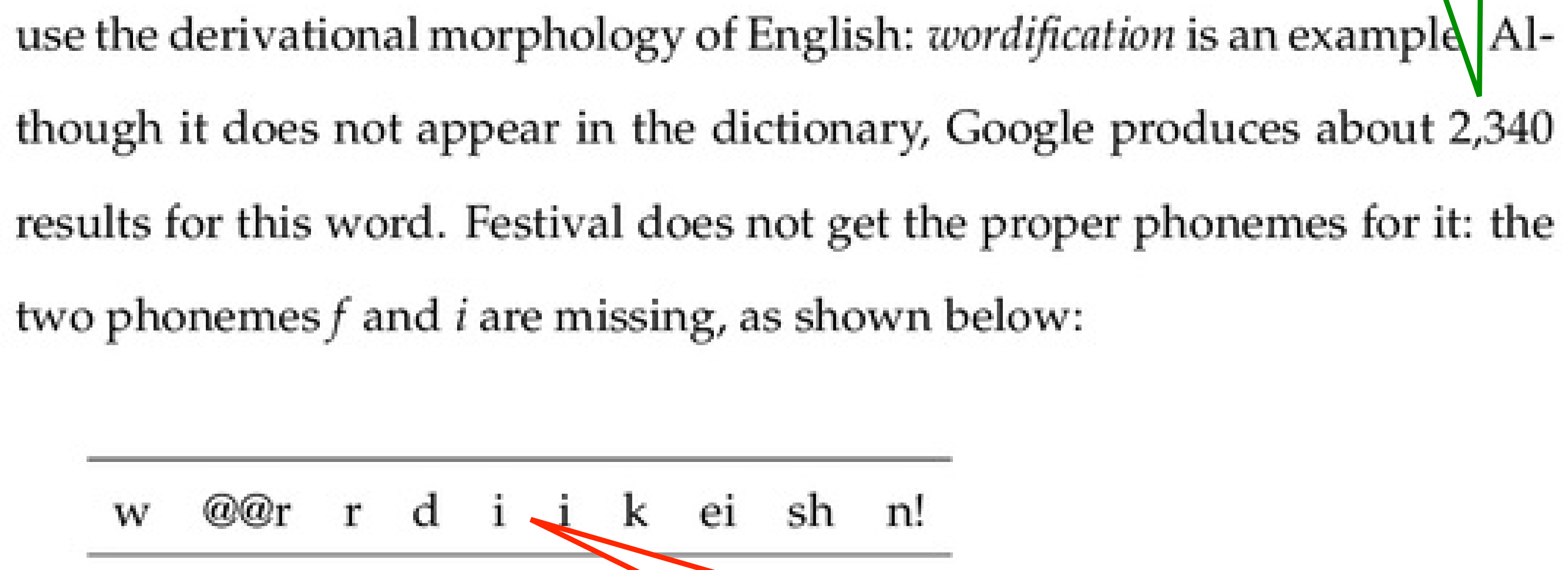
given speaker of a given accent, e.g. long first ‘oo’ in ‘doctor’ (id_83). The phonetic structure of non-dictionary words are guessed based on English 10 Letter-To-Sound (LTS) rules. These rules can be listed, and used by a CART tree, in concordance with letter context, to make a guess

correctly says that LTS is implemented as a CART

slightly poor wording: the CART **is** the letter-to-sound model; it doesn’t “use” LTS rules, it **is** those rules

Lexicon, LTS

good use of empirical data



use the derivational morphology of English: *wordification* is an example. Although it does not appear in the dictionary, Google produces about 2,340 results for this word. Festival does not get the proper phonemes for it: the two phonemes *f* and *i* are missing, as shown below:

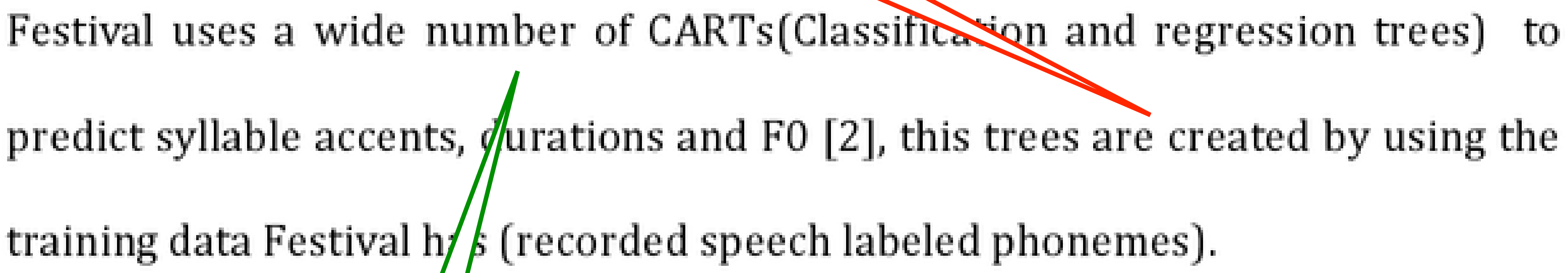
w	@@r	r	d	i	i	k	ei	sh	n!

nice compact format

but **where** should the missing phonemes be? should have annotated the phonetic string to indicate this

too little information on
exactly how CARTS are configured
(what are the predictors?)

on, LTS



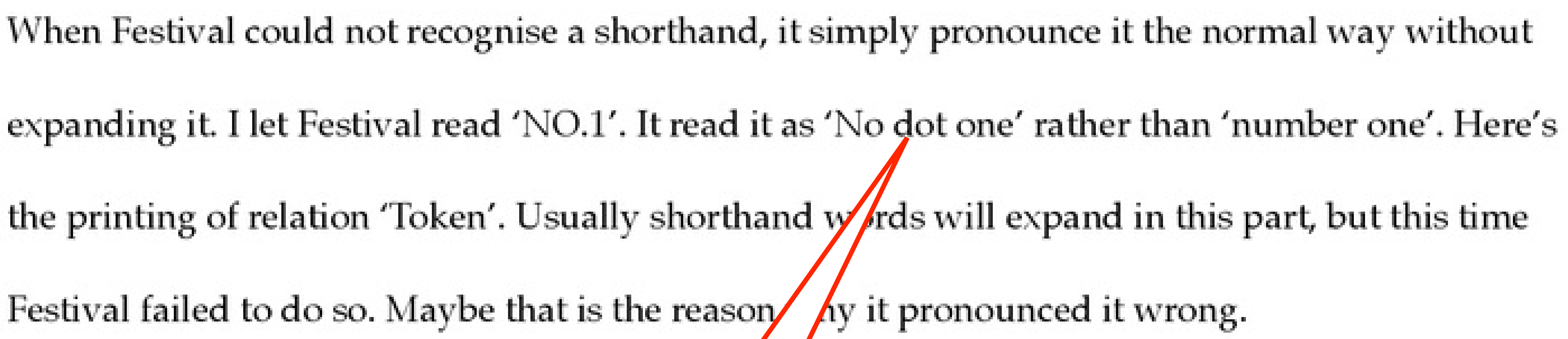
Festival uses a wide number of CARTs(Classification and regression trees) to predict syllable accents, durations and F0 [2], this trees are created by using the training data Festival has (recorded speech labeled phonemes).

correctly says that CARTs are used
for various tasks

not all training data is in the form of
labelled speech

Lexicon, LTS

- Pronunciation

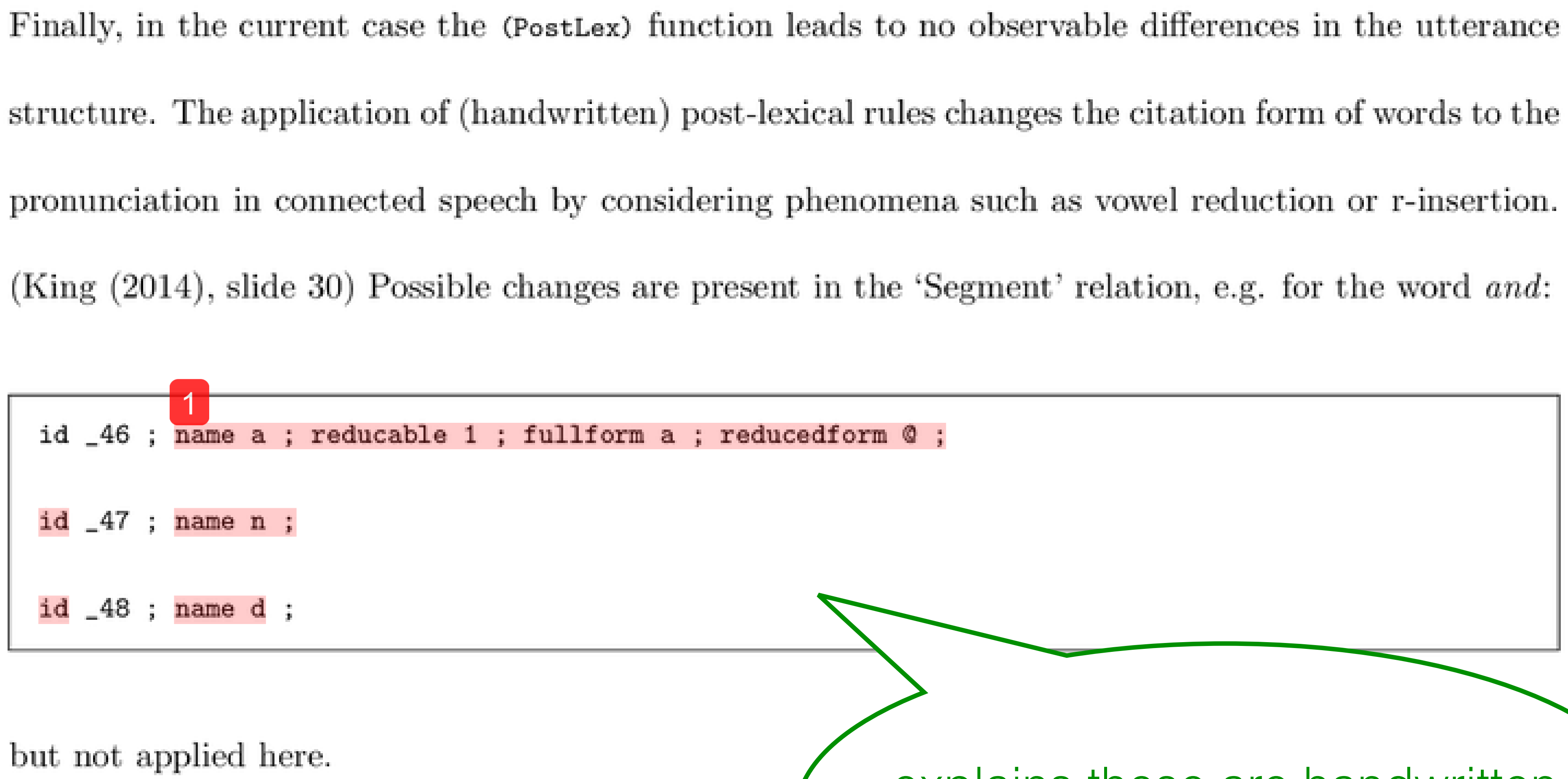


When Festival could not recognise a shorthand, it simply pronounce it the normal way without expanding it. I let Festival read 'NO.1'. It read it as 'No dot one' rather than 'number one'. Here's the printing of relation 'Token'. Usually shorthand words will expand in this part, but this time Festival failed to do so. Maybe that is the reason why it pronounced it wrong.

the writer even said as much!

a common mistake: a pronunciation problem was caused by an earlier problem (e.g., in tokenisation) - so this is **not** an error in the pronunciation module

Postlexical rules



Finally, in the current case the (PostLex) function leads to no observable differences in the utterance structure. The application of (handwritten) post-lexical rules changes the citation form of words to the pronunciation in connected speech by considering phenomena such as vowel reduction or r-insertion. (King (2014), slide 30) Possible changes are present in the 'Segment' relation, e.g. for the word *and*:

```
id _46 ; name a ; reducable 1 ; fullform a ; reducedform @ ;

id _47 ; name n ;

id _48 ; name d ;
```

but not applied here.

explains these are handwritten rules, and gives example rules are used, not a lexicon

postlexical rules do more than just possessive “ ’s ”

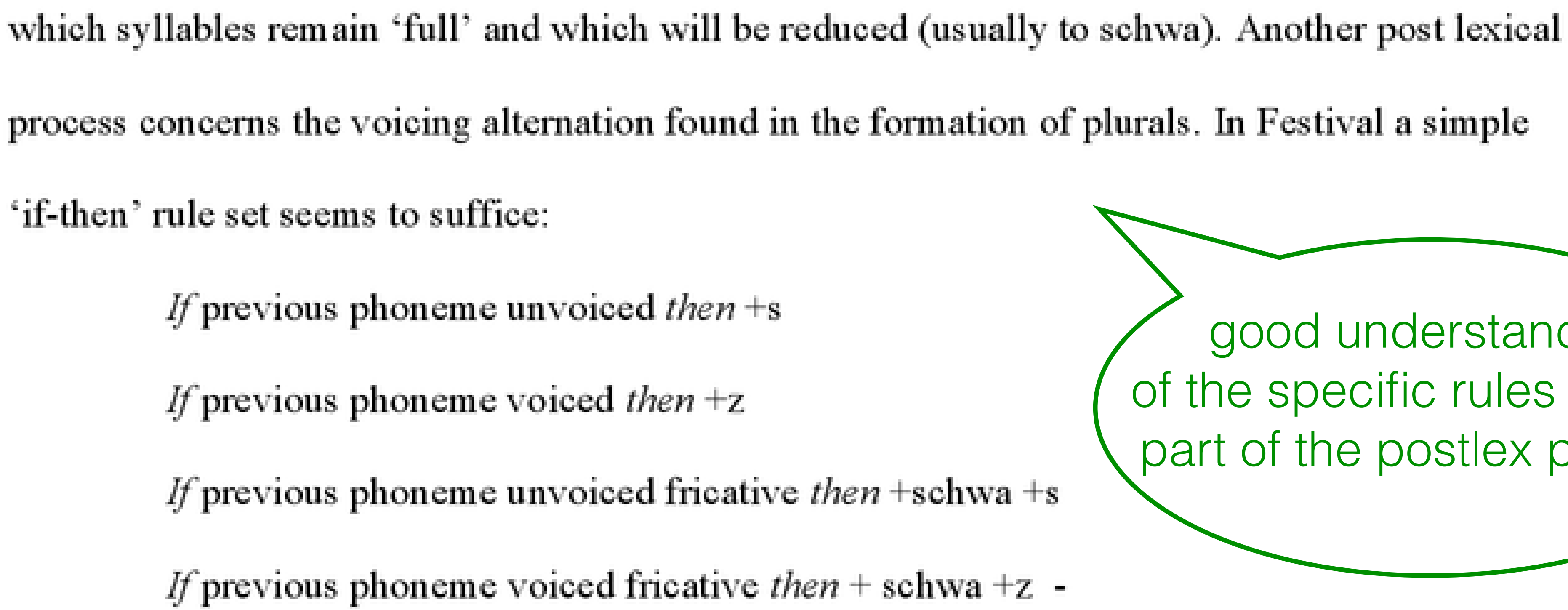
Postlexical rules

4.9 PostLex -This command uses a lexicon to analyze possessive s “‘s” contained in the words array. When it identifies a possessive ‘s it will delete the swha sound, it will remove the shwa and add fricative sound to the end of the previous word, e.g. “Samantha’s”

Before (PostLex):	After (PostLex)
Samantha	Samantha
s.@	s.@
m.a.n (1)	m.a.n (1)
th.@	th.@.z
's	[illegible]
@.z	[illegible]

good example given, nicely laid out in a table

Postlexical rules



which syllables remain 'full' and which will be reduced (usually to schwa). Another post lexical process concerns the voicing alternation found in the formation of plurals. In Festival a simple 'if-then' rule set seems to suffice:

If previous phoneme unvoiced *then* +s

If previous phoneme voiced *then* +z

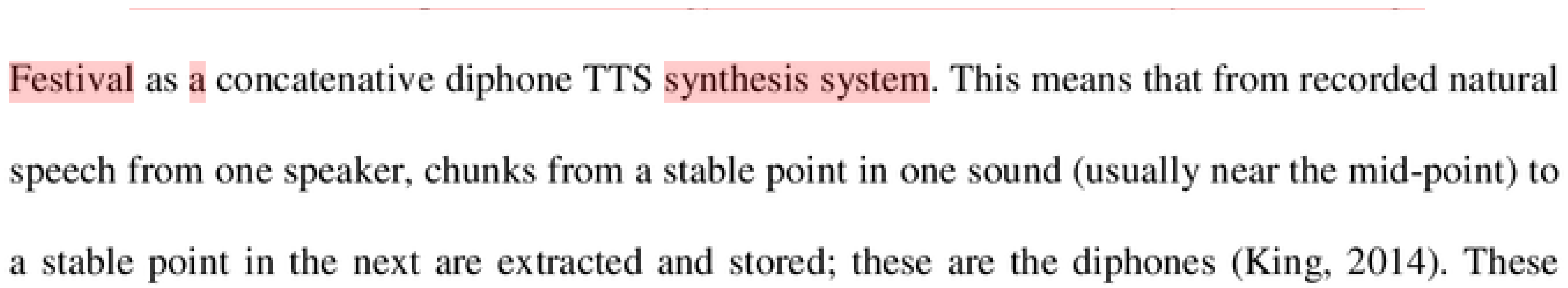
If previous phoneme unvoiced fricative *then* +schwa +s

If previous phoneme voiced fricative *then* + schwa +z -

good understanding of the specific rules used in part of the postlex process

could have cited the source of this information

Waveform generation

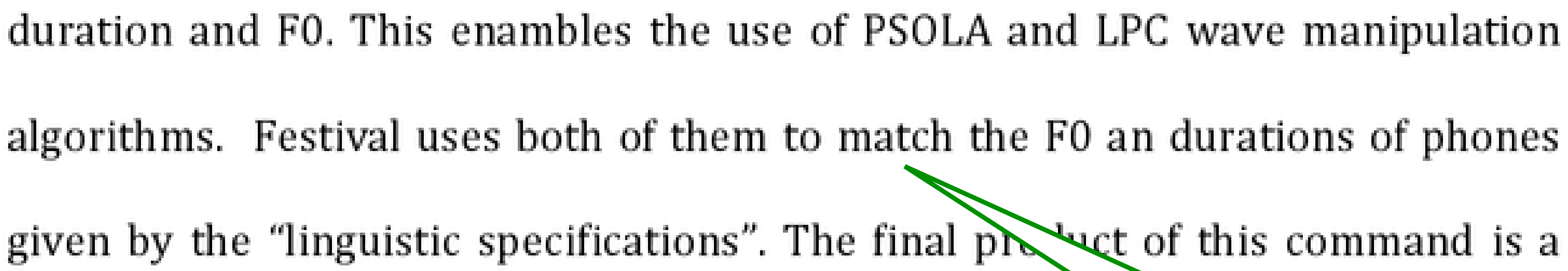


Festival as a concatenative diphone TTS synthesis system. This means that from recorded natural speech from one speaker, chunks from a stable point in one sound (usually near the mid-point) to a stable point in the next are extracted and stored; these are the diphones (King, 2014). These

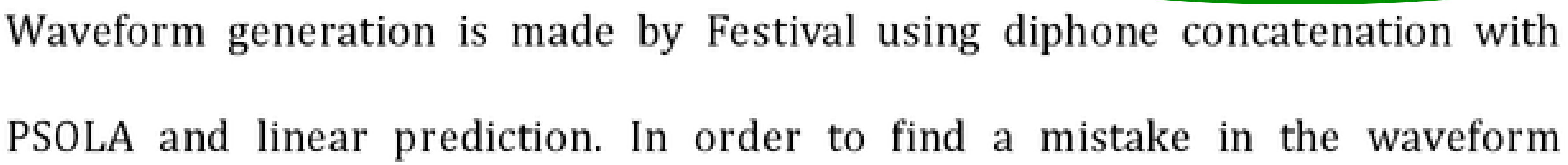
compact definition of diphones - demonstrates understanding of why they are used

Waveform generation

mentions techniques


duration and F0. This enambles the use of PSOLA and LPC wave manipulation algorithms. Festival uses both of them to match the F0 an durations of phones given by the "linguistic specifications". The final product of this command is a

says why they are needed


Waveform generation is made by Festival using diphone concatenation with PSOLA and linear prediction. In order to find a mistake in the waveform

but no detail on how they work, or what the limitations are

Waveform generation

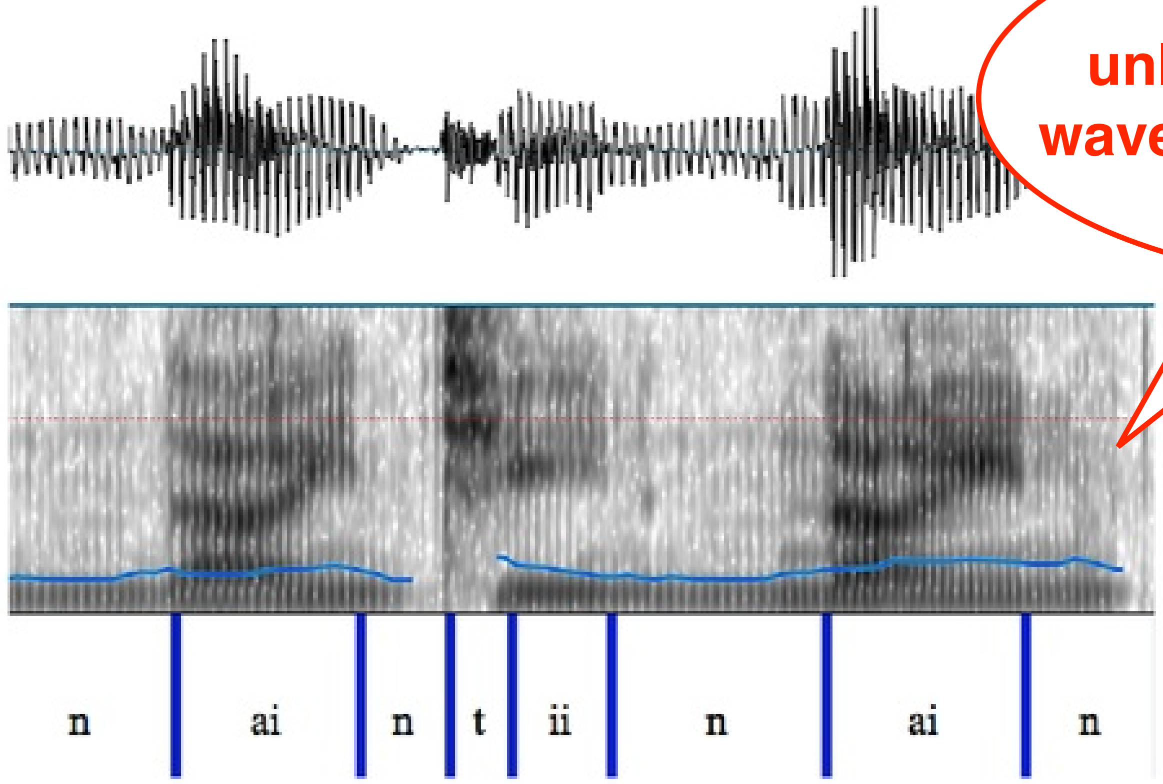


Figure 1: Spectrogram detail for *nineteen nine[ty]*

unlabelled axes on both waveform and spectrogram

nicely annotated with phones

Waveform generation

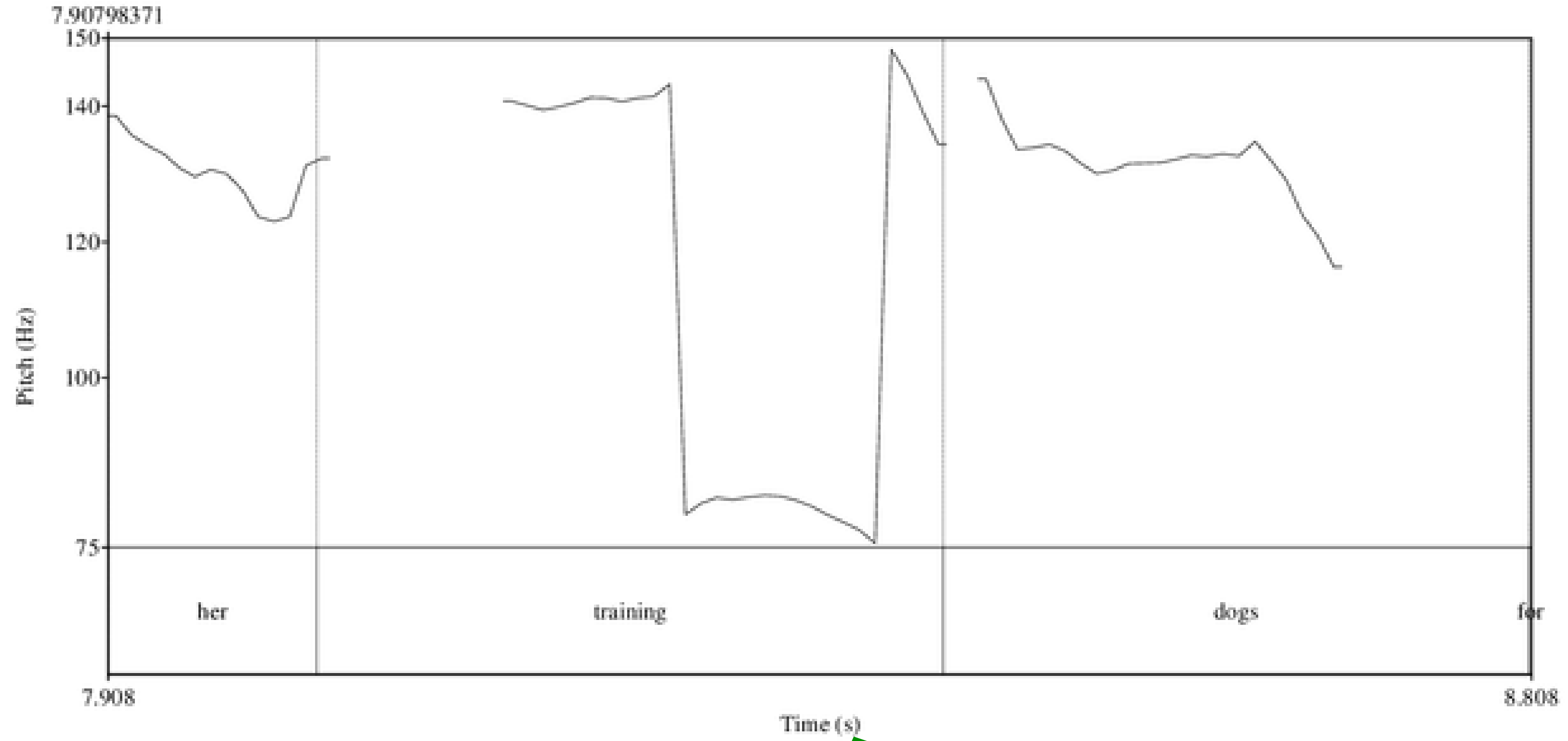


Figure 2: Pitch trace of ‘her training dogs’ with clear ri[illegible] in ‘training’ and falling in ‘dogs’, and a jumpy, unnatural pitch in the second half of ‘training’.

descriptive caption

axes labelled
(although font too small)

Waveform generation

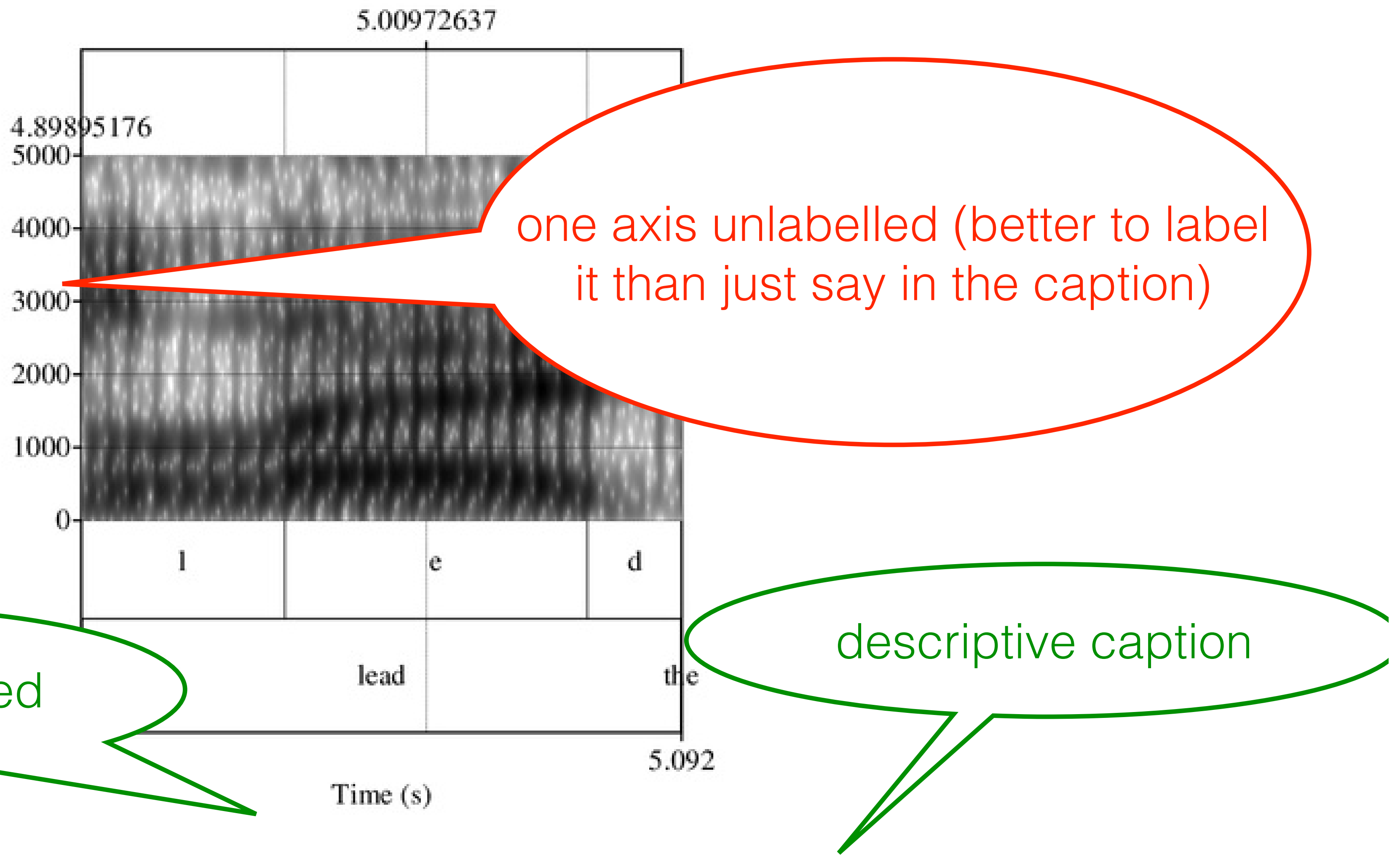


Figure 4: Spectrogram of 'led'. Formant values read at time-point indicated (approx.. 5s), frequency scale is indicated on the left-hand side. 'e': F1 = 594 Hz, F2 = 1629 Hz.

Waveform generation

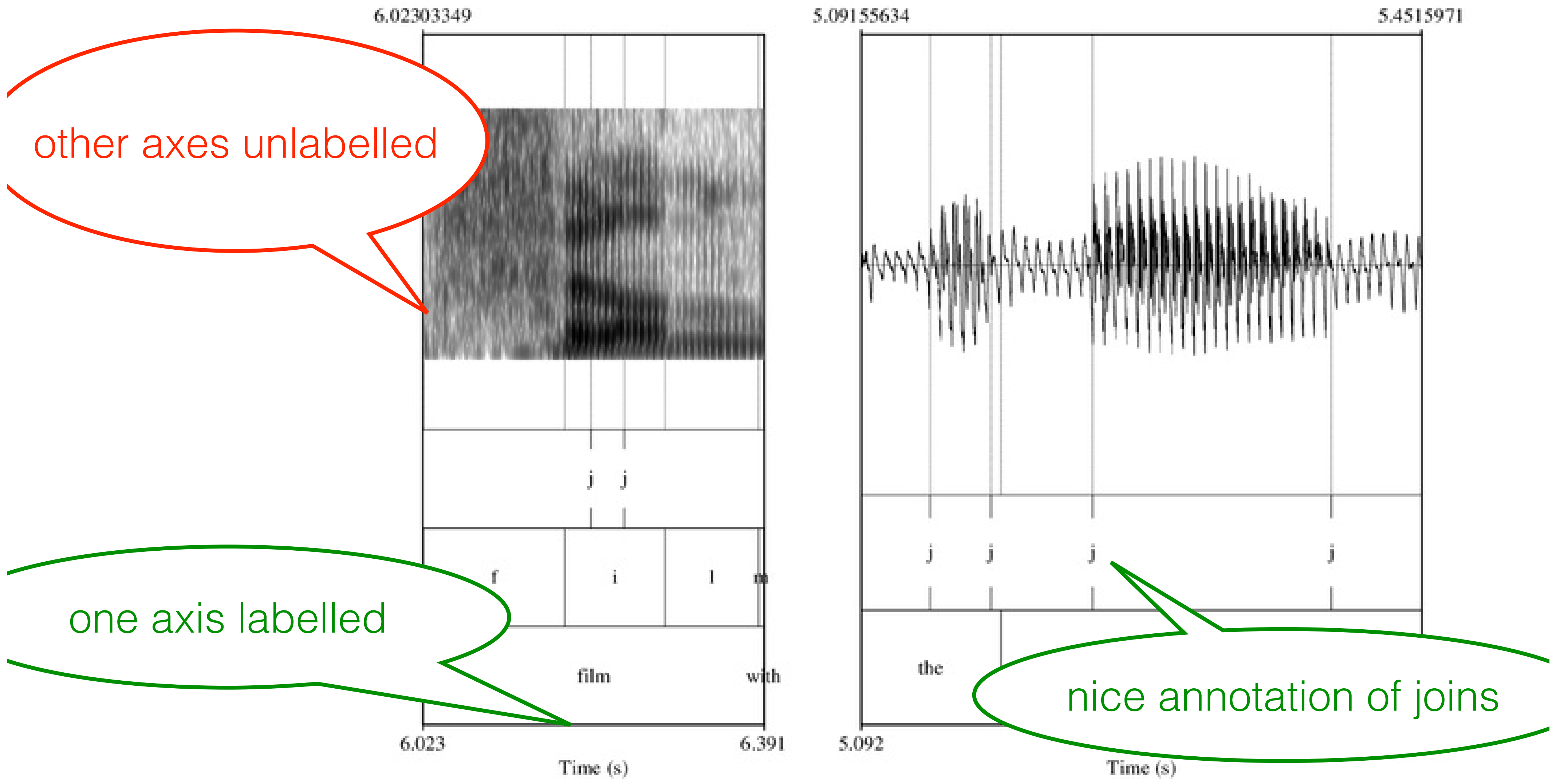


Figure 7: Joins indicated on a spectrogram of 'film' and waveform of 'the men', respectively.

Waveform generation

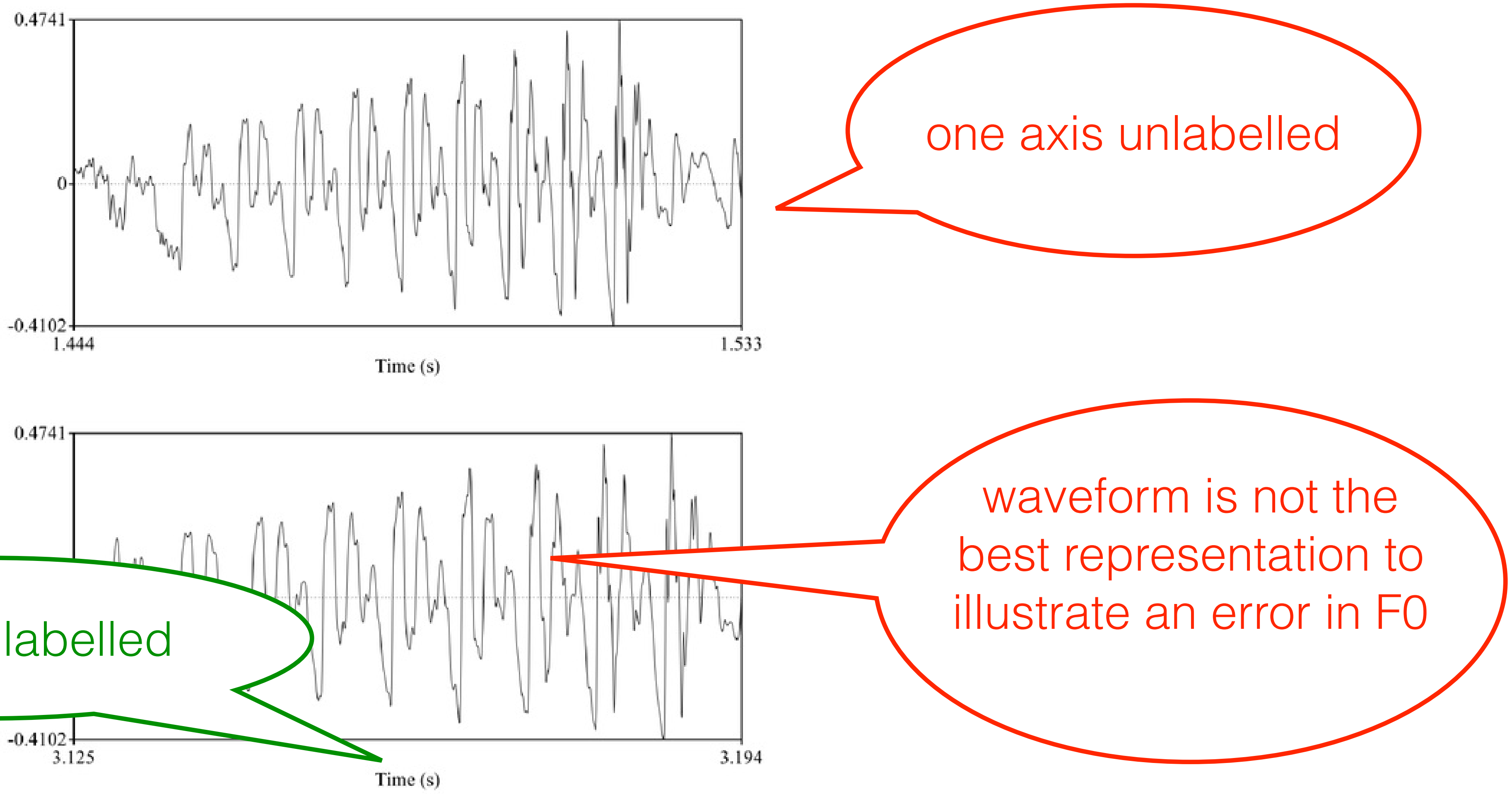
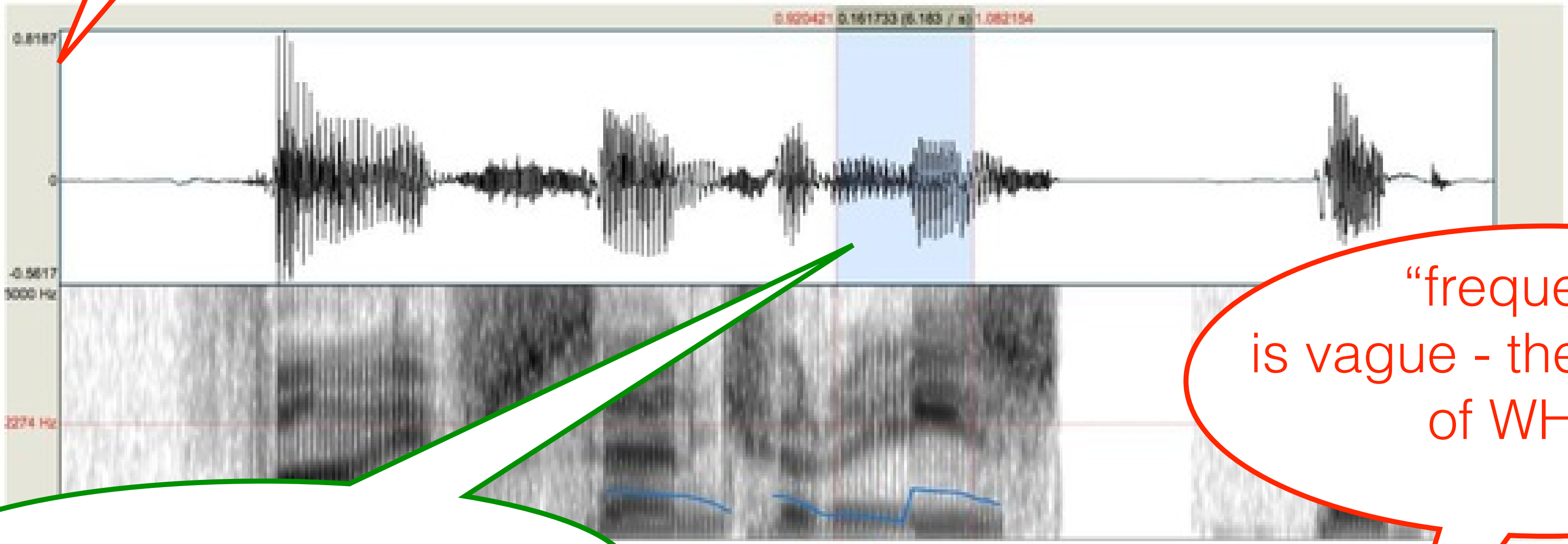


Figure 4. The top illustration is a waveform of sentence (o) without focus of the sentence on a syllable "grey". The bottom illustration is a waveform of sentence (p) with focus of the sentence on a syllable "grey". There are no notable differences between two waveforms.

aveform generation

unlabelled axes



highlights area of interest

"frequency" is vague - the frequency of WHAT?

On the highlighted area of the phrase, a clear variation on the Frequency can be seen, nonetheless, that part corresponds to the syllable "ries" of centuries, and there shouldn't be any pitch variation. Also when playing the sound the word

Waveform generation

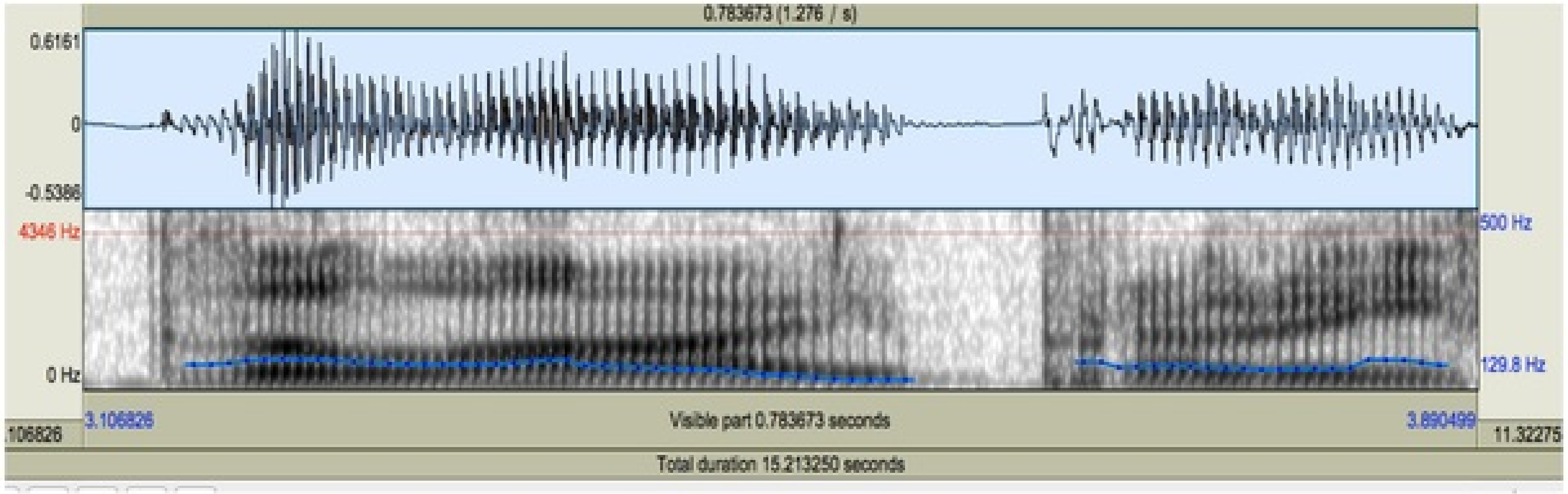


Figure iv: *Looking at the spectral envelope in the spectrogram, we can see that not only is there an unnaturally long duration, but a dark band created by the clear splicing together of two incompatible diphones indicating a sudden increase in pressure which has created a synthetic clicking sound, degrading the naturalness of the sound.*

good caption

but no idea **where** on this spectrogram we can see this

Waveform generation

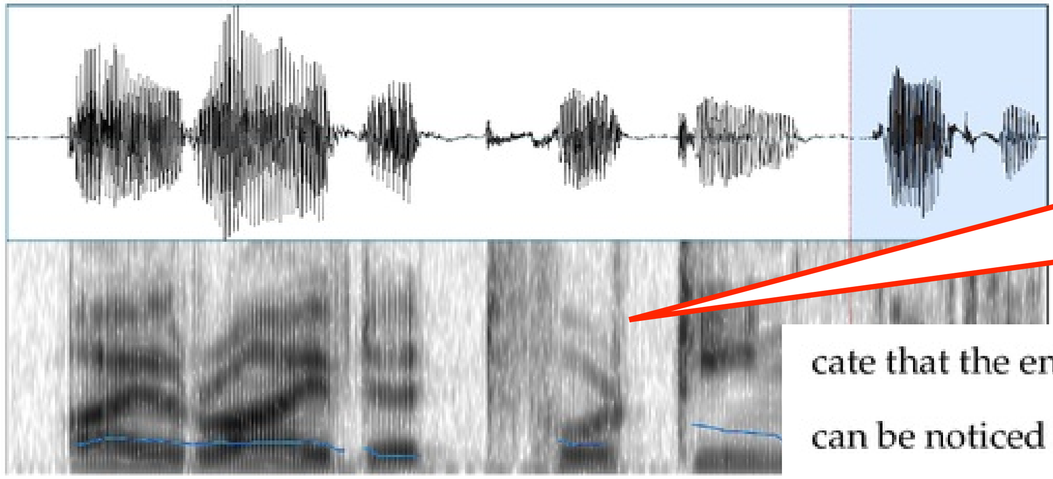
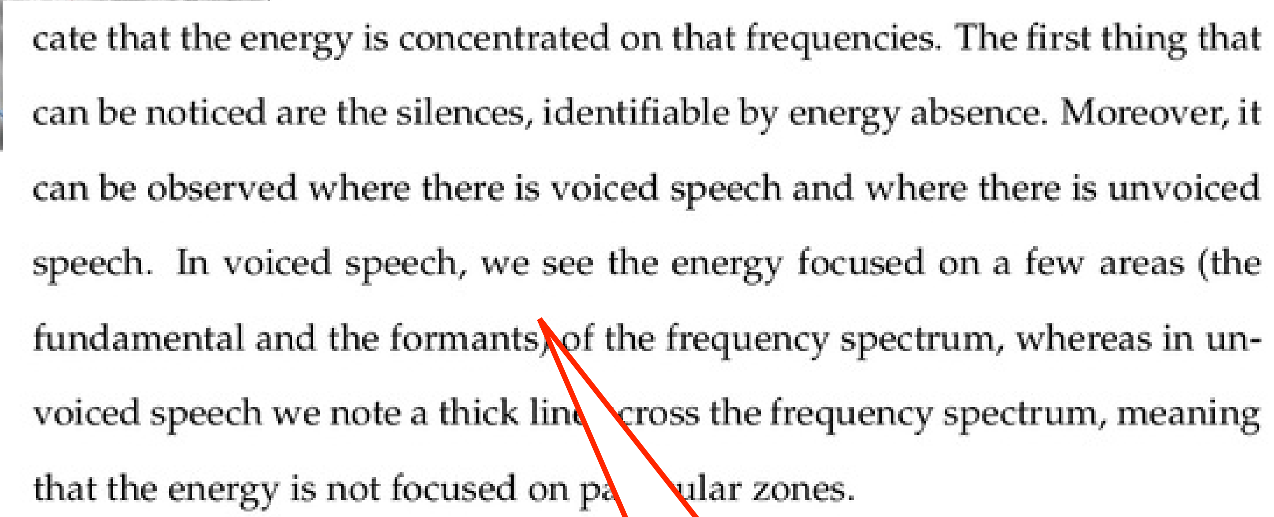


Figure 1: Output waveform



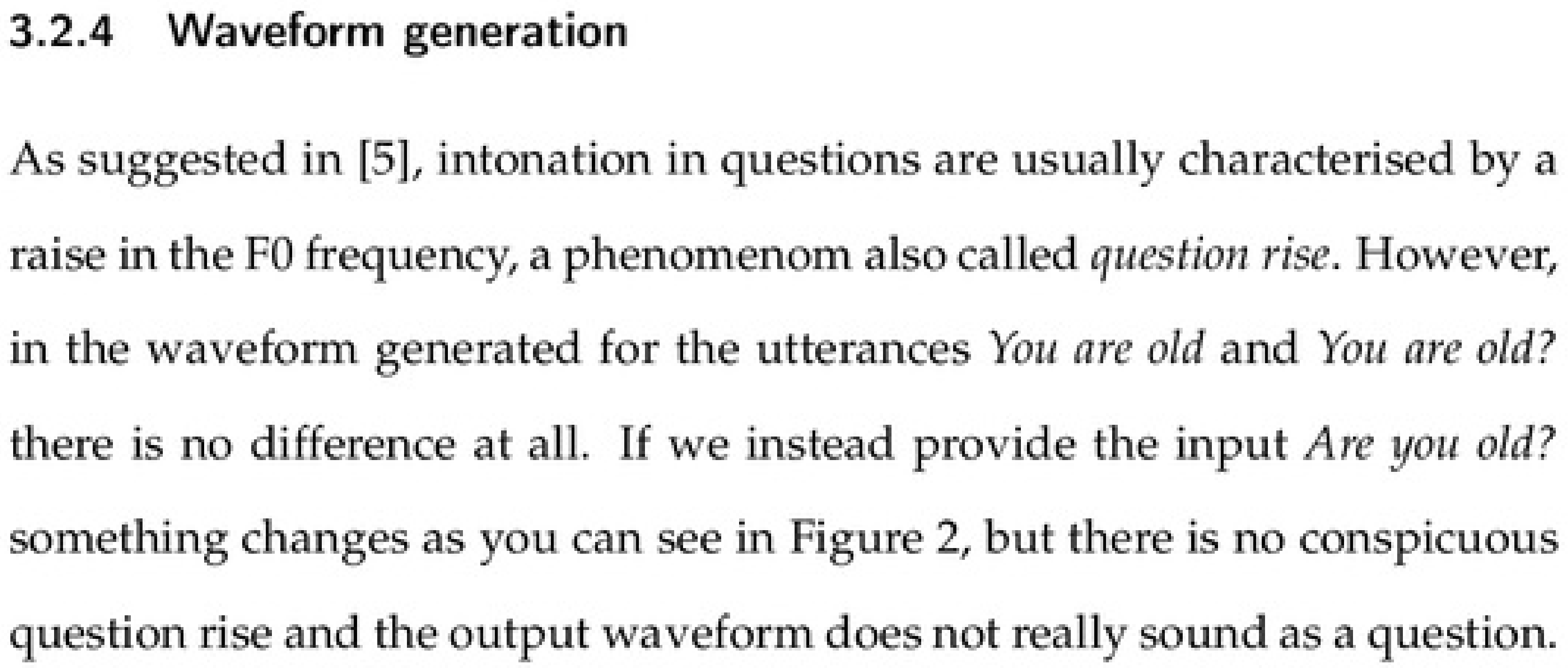
cate that the energy is concentrated on that frequencies. The first thing that can be noticed are the silences, identifiable by energy absence. Moreover, it can be observed where there is voiced speech and where there is unvoiced speech. In voiced speech, we see the energy focused on a few areas (the fundamental and the formants) of the frequency spectrum, whereas in unvoiced speech we note a thick lin[illegible]cross the frequency spectrum, meaning that the energy is not focused on pa[illegible]ular zones.

should have **annotated** the spectrogram

poor caption - uninformative

no idea **where** on the spectrogram we can see the things mentioned

Waveform generation

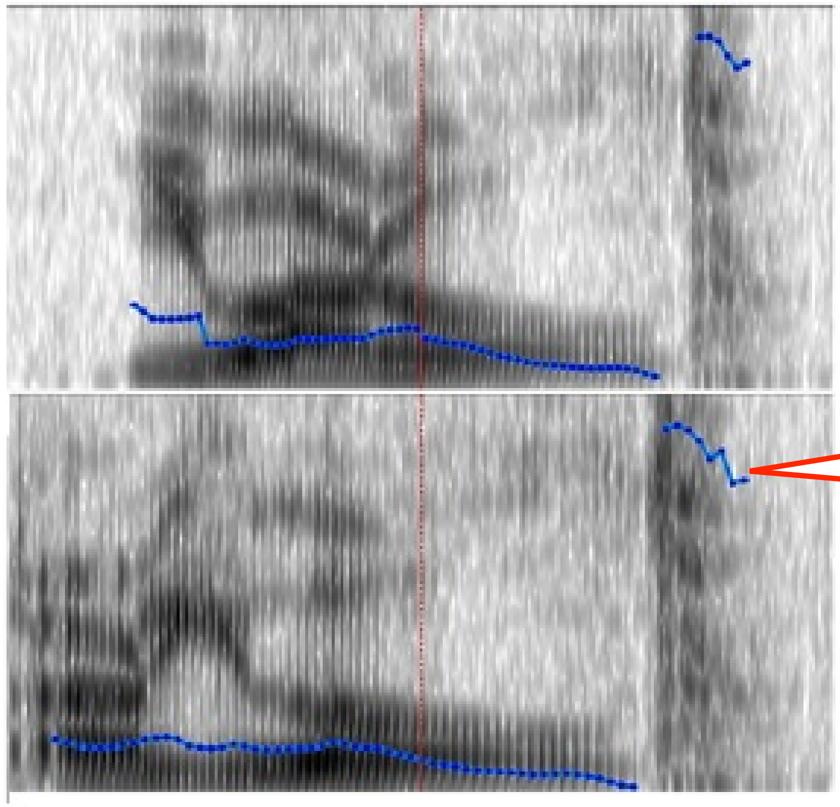


3.2.4 Waveform generation

As suggested in [5], intonation in questions are usually characterised by a raise in the F0 frequency, a phenomenom also called *question rise*. However, in the waveform generated for the utterances *You are old* and *You are old?* there is no difference at all. If we instead provide the input *Are you old?* something changes as you can see in Figure 2, but there is no conspicuous question rise and the output waveform does not really sound as a question.

not clearly demonstrated that this is a waveform error - **should** have shown that the linguistic specification differed between these two sentences

Waveform generation



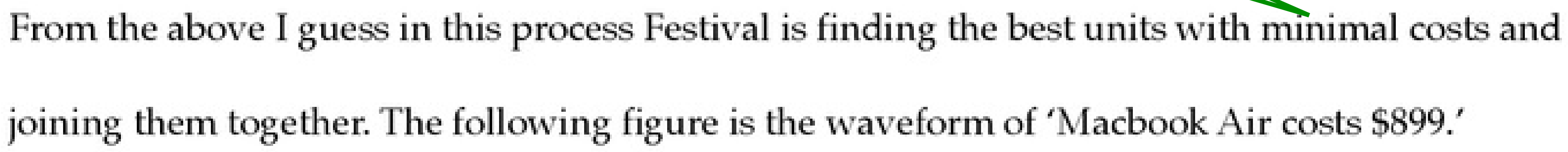
unlabelled axes

what is the blue line?

Figure 2: Spectogram for utterance *You are old* (above) and *Are you old?* (below)

Waveform generation

yes, Festival is minimising some cost functions

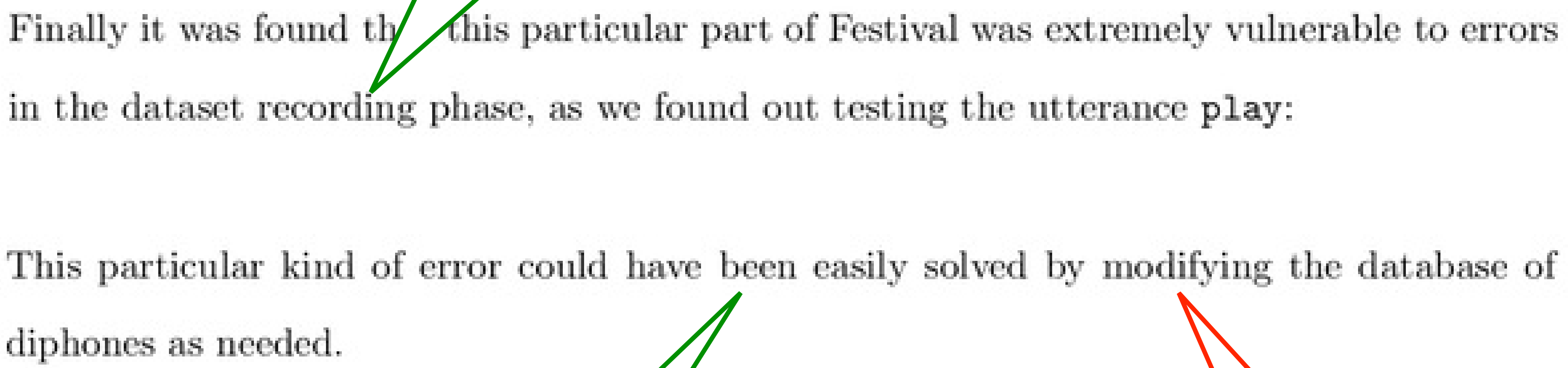


From the above I guess in this process Festival is finding the best units with minimal costs and joining them together. The following figure is the waveform of ‘Macbook Air costs $899.’

but **what are they**? just a couple of additional sentences would get several more marks

Suggesting improvements

correct diagnosis of a source of errors



Finally it was found th this particular part of Festival was extremely vulnerable to errors in the dataset recording phase, as we found out testing the utterance `play`:

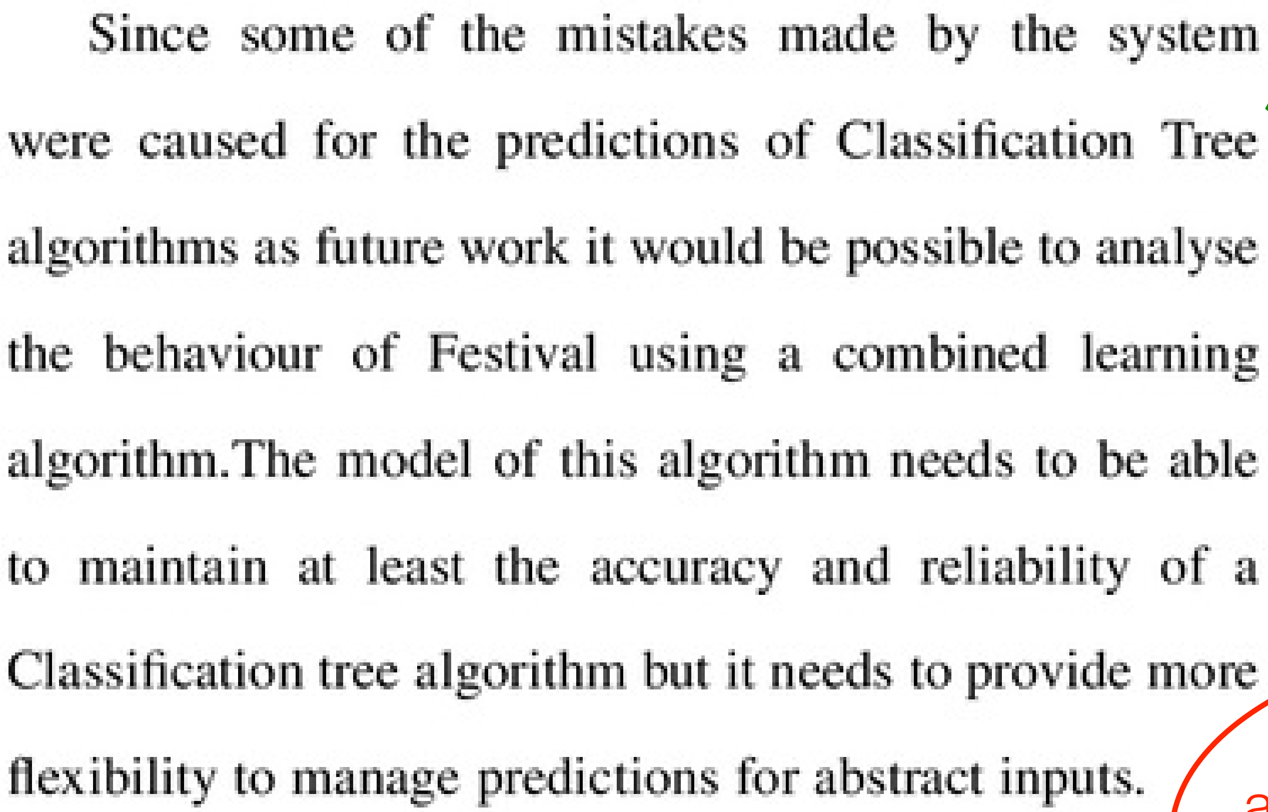
This particular kind of error could have been easily solved by modifying the database of diphones as needed.

correct solution

but **should** have specified exactly HOW the database should be modified

Suggesting improvements

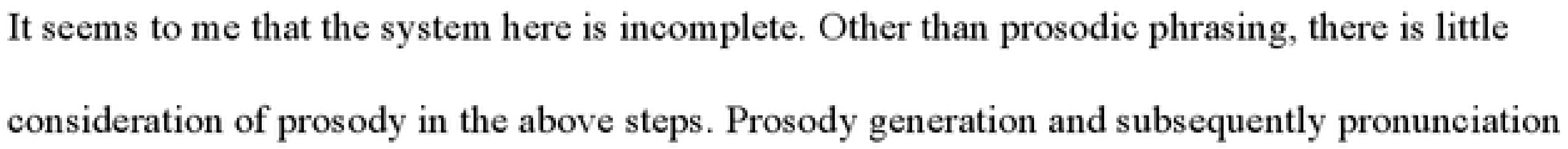
yes, CARTs make mistakes



Since some of the mistakes made by the system were caused for the predictions of Classification Tree algorithms as future work it would be possible to analyse the behaviour of Festival using a combined learning algorithm.The model of this algorithm needs to be able to maintain at least the accuracy and reliability of a Classification tree algorithm but it needs to provide more flexibility to manage predictions for abstract inputs.

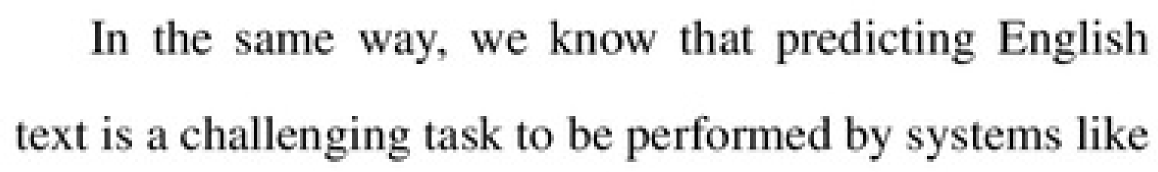
but this is vague and not specific to any particular part of the system

Suggesting improvements


It seems to me that the system here is incomplete. Other than prosodic phrasing, there is little consideration of prosody in the above steps. Prosody generation and subsequently pronunciation

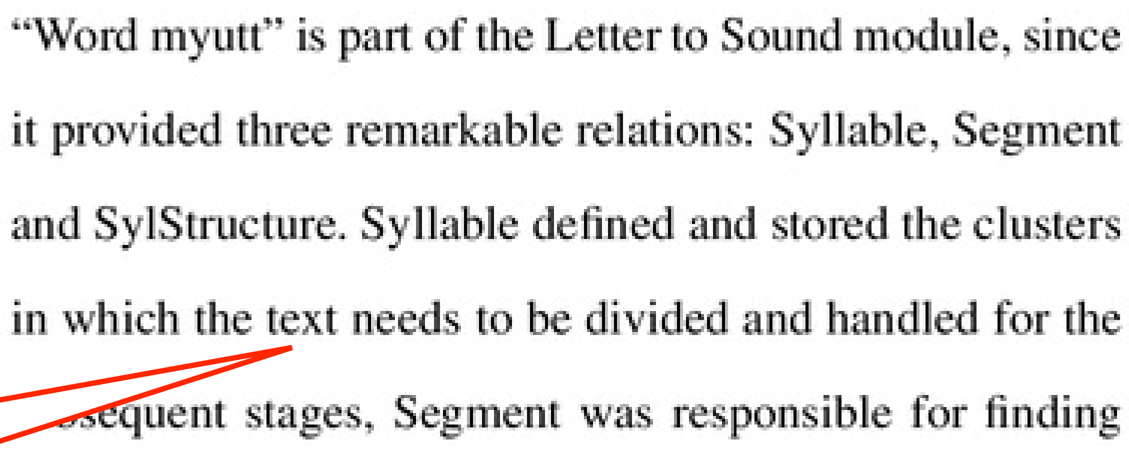
correctly states
that prosody prediction is
somewhat limited

Vague or imprecise wording



In the same way, we know that predicting English text is a challenging task to be performed by systems like

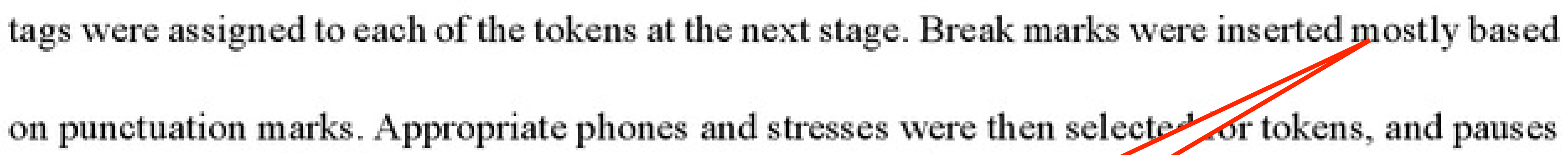
Festival does not "predict text"



"Word myutt" is part of the Letter to Sound module, since it provided three remarkable relations: Syllable, Segment and SylStructure. Syllable defined and stored the clusters in which the text needs to be divided and handled for the subsequent stages, Segment was responsible for finding

Syllables are a phonological structure in Festival, not text

Vague or imprecise wording


tags were assigned to each of the tokens at the next stage. Break marks were inserted mostly based on punctuation marks. Appropriate phones and stresses were then selected for tokens, and pauses

by “mostly” do you mean “only” ?

How to do better next time

- **Keep a copy** of your marked assignment for future reference, including the marking and feedback sheets
  - PGs - you are allowed to keep the original
  - UGs - make a copy or scan, then return the original
- **Compare** your first assignment with these feedback slides
  - add your own feedback - *“notes to self”*
- **Read** the lab handouts and the forum in Learn
- Carefully **study** the structured marking sheet
- Draft your second assignment, then **compare** that to these slides