ISSN 2033-3544

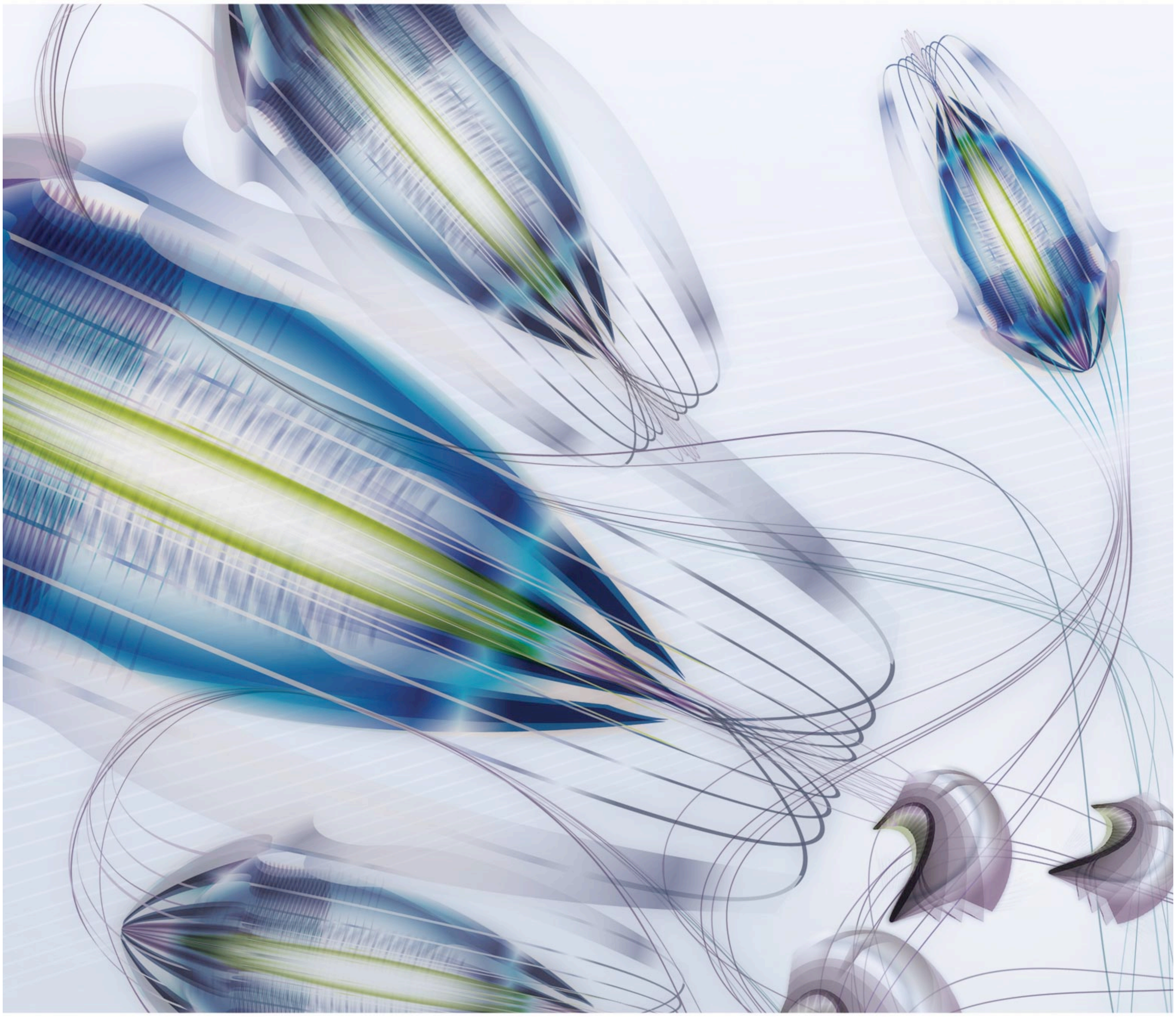
CLiPS

COMPUTATIONAL LINGUISTICS & PSYCHOLINGUISTICS
TECHNICAL REPORT SERIES, CTRS-004, JULY 2013

WWW.CLIPS.UA.AC.BE/CTRS

STYLENE: an environment for Stylometry and Readability Research for Dutch

Walter Daelemans, Véronique Hoste



Universiteit Antwerpen

COMPUTATIONAL LINGUISTICS & PSYCHOLINGUISTICS RESEARCH CENTER CLiPS

CTRS-004
July 19, 2013

STYLENE: an Environment for Stylometry and Readability Research for Dutch

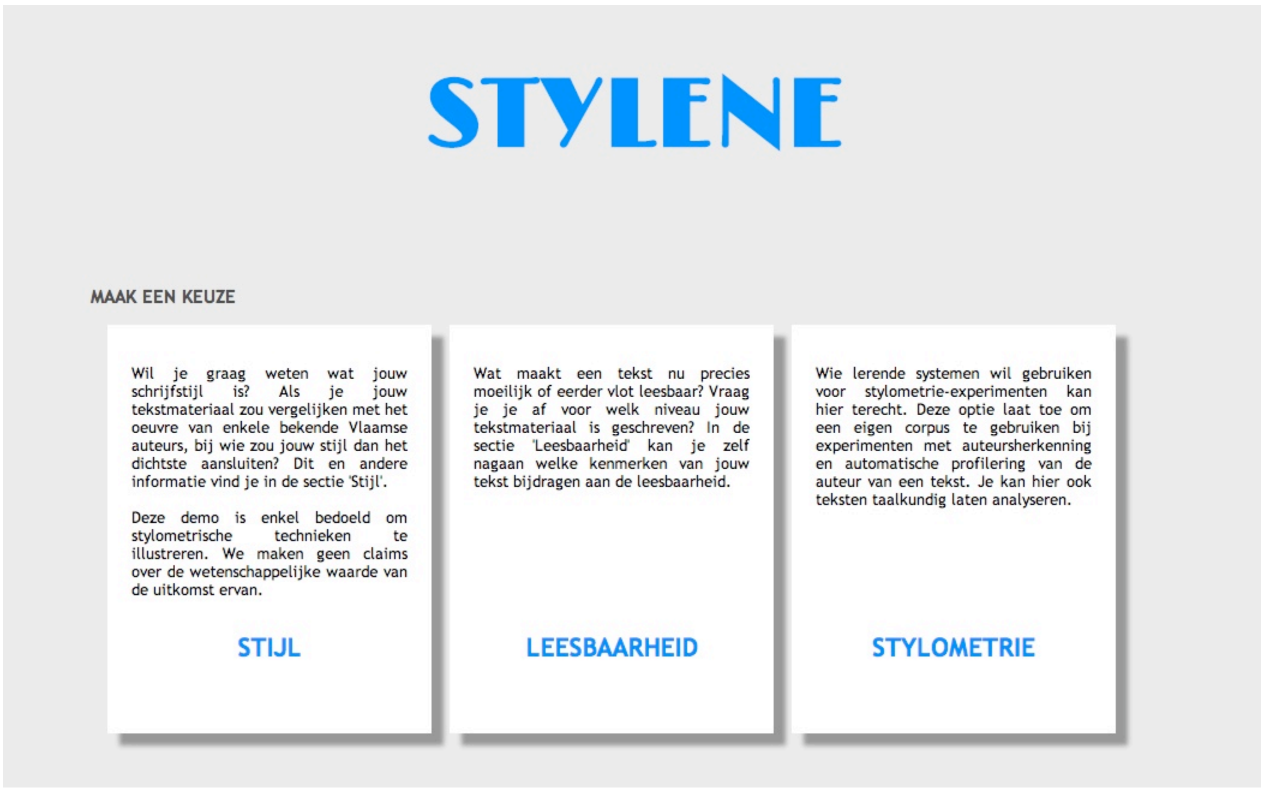
Walter Daelemans
Computational Linguistics & Psycholinguistics Research Center
University of Antwerp
walter.daelemans@ua.ac.be

Véronique Hoste
LT3 Language and Translation Technology Team
University College Ghent / Ghent University
veronique.hoste@hogent.be

http://www.clips.ua.ac.be/projects/stylene

STYLENE: an Environment for Stylometry and Readability Research for Dutch

This report provides a practical introduction to the use of the interface and corresponding backend system that was developed in the context of the Stylene project. The goal of that project was to implement a robust, modular system for stylometry and readability research on the basis of existing methods, and the development of a web service that allows researchers in the humanities and social sciences to analyze texts with this system.


STYLENE

MAAK EEN KEUZE

Wil je graag weten wat jouw schrijfstijl is? Als je jouw tekstmateriaal zou vergelijken met het oeuvre van enkele bekende Vlaamse auteurs, bij wie zou jouw stijl dan het dichtste aansluiten? Dit en andere informatie vind je in de sectie 'Stijl'.

Deze demo is enkel bedoeld om stylometrische technieken te illustreren. We maken geen claims over de wetenschappelijke waarde van de uitkomst ervan.

STIJL

Wat maakt een tekst nu precies moeilijk of eerder vlot leesbaar? Vraag je je af voor welk niveau jouw tekstmateriaal is geschreven? In de sectie 'Leesbaarheid' kan je zelf nagaan welke kenmerken van jouw tekst bijdragen aan de leesbaarheid.

LEESBAARHEID

Wie lerende systemen wil gebruiken voor stylometrie-experimenten kan hier terecht. Deze optie laat toe om een eigen corpus te gebruiken bij experimenten met auteursherkenning en automatische profilering van de auteur van een tekst. Je kan hier ook teksten taalkundig laten analyseren.

STYLOMETRIE

WEBSITE: http://www.stylene.be

Preface

The last decade has seen an increase in research on computational stylometry, the subarea of natural language processing that concerns itself with the categorization of texts according to psychological and sociological properties of their authors (Daelemans, 2013). Also called text profiling, this research tries to develop systems, mostly based on text analytics techniques, that combine natural language processing and machine learning methods. These systems are trained to determine whether the author of a text is male or female, her education level, region of origin, personality, and even mental health, whether she is a native speaker or not, and many other potentially useful attributes. Of course, authorship attribution research has existed for a long time, and is in a sense the limit case of computational stylometry: supposing that everyone has a unique idiolect or "stylome" (Van Halteren et al, 2005), it should be possible to assign texts to specific authors for whom models of their stylome exist.

Another useful type of information that can be extracted from text using the text analytics approach is the readability of the text. Readability research and the development of automatic techniques to determine text readability already dates back to the early years of the last century. However, due to the progress made in domains such as natural language processing and machine learning, new readability prediction techniques are starting to be developed which not only take into account superficial text characteristics such as word and sentence length. Current approaches model lexical, syntactic, semantic and discourse complexity, while also taking into account shallow traditional text characteristics. Furthermore, the focus has shifted from using the formulas to select reading material for children or L2 language learners to assessing the readability of a variety of text types with other user groups or applications in mind.

This report provides a practical introduction to the use of the interface and corresponding backend system that was developed in the context of the Stylene (Stylometrie en Leesbaarheid voor het Nederlands; An environment for stylometry and readability research for Dutch) project. The goal of that project was to implement a robust, modular system for stylometry and readability research on the basis of existing methods, and the development of a web service that allows researchers in the humanities and social sciences to analyze texts with this system. The project was funded by the Flemish ministry for Economics, Science, and Innovation (EWI). The system was developed over 2010-2012 in a cooperation between the CLiPS[1] and LT3[2] research groups, and was coordinated by Walter Daelemans. The interface can be found at: http://www.stylene.be

There are three sub-interfaces: (i) a popularization interface intended to provide basic insight in what stylometry can do; (ii) a readability interface that allows input of texts and provides elementary and more advanced feedback on the readability of the text; (iii) a machine learning interface that allows basic experiments in computational stylometry.

In this report we will describe the use of the different interfaces and some of the underlying methods and approaches. In appendix we also provide documentation about the stand-alone system for machine learning based stylometry that underlies the third interface, but that allows more options and flexibility as a stand-alone system than can be accessed from the interface.

[1] http://www.clips.ua.ac.be
[2] http://lt3.hogent.be/en/

ACKNOWLEDGEMENTS

Several people participated in the development of the interface. We gratefully acknowledge their contributions. Herwig De Smet implemented most of the interface. He also implemented the stand-alone stylometry system underlying the machine learning interface and wrote the manual for this stand-alone version (provided here as an appendix). Koen Vereeken implemented a genetic algorithm optimization system used to develop the models. Guy De Pauw solved various server-related issues. Vincent Van Asch implemented the stylometry popularization interface and the simplified models used in it. Frederik Vaassen and Claudia Peersman stress-tested different versions of the system. Peter Velaerts implemented the various classical formulas in the readability prediction interface. Orphée De Clercq implemented the different flavors of the Hendi readability metric.

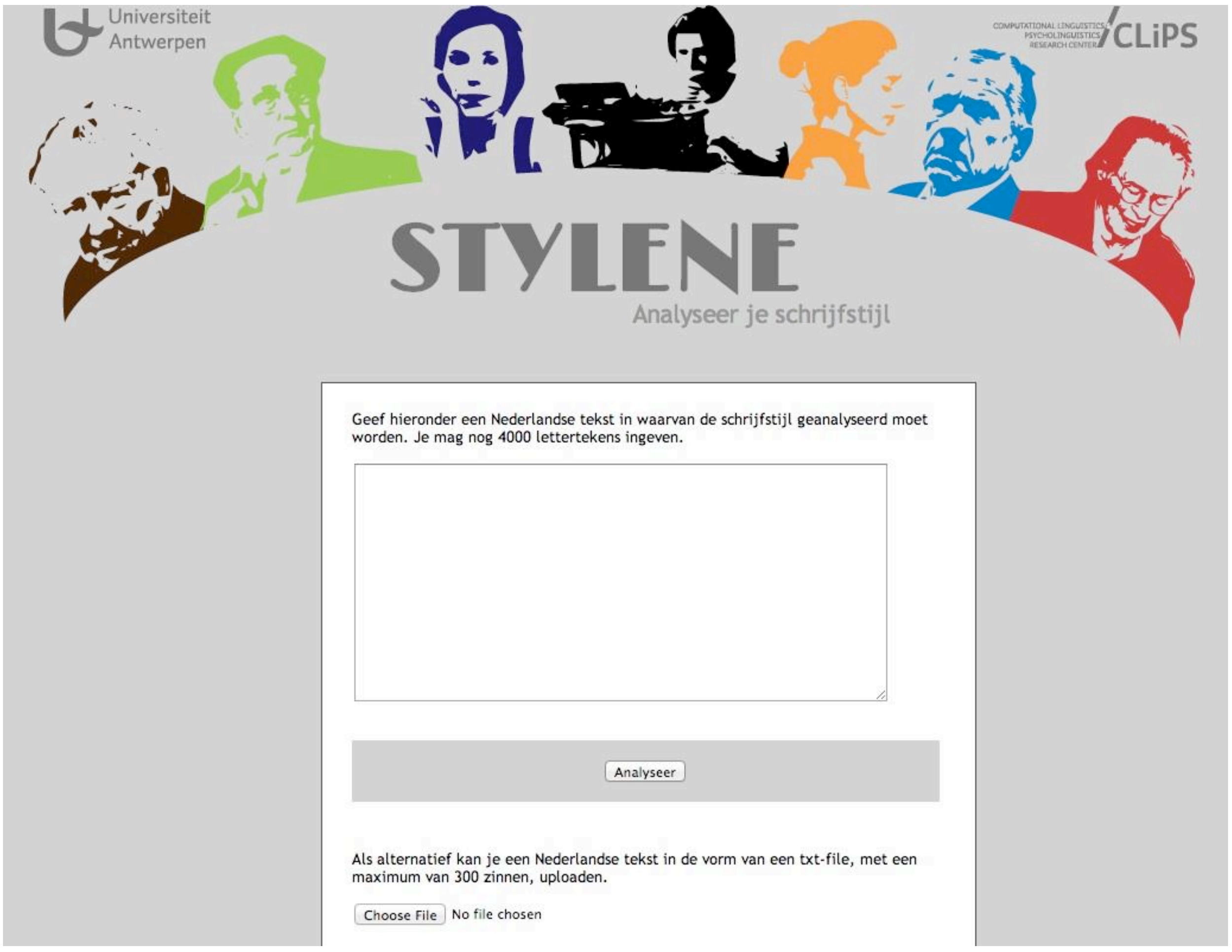
More information about the Stylene project can be obtained from:

Prof. Dr. Walter Daelemans
CLiPS, Department of Linguistics, University of Antwerp
walter.daelemans@ua.ac.be

Table of contents

1 The Stylometry Popularization Interface

Computational Stylometry is not yet well-known outside computational linguistics and the specialized digital humanities research. In order to educate interested lay persons and humanities colleagues about the possibilities (and limitations) of the approach, an interface was designed at CLiPS by Walter Daelemans and Vincent Van Asch and developed and implemented by the latter to help a general audience understand computational stylometry in an easy and fun way. An early version was tested out successfully during the 2011 “Wetenschapsweek” (Science Week) with secondary school pupils, and afterwards extended. There has been a large interest for the interface (around 50 visitors per month) and quite some media attention. Figure 1 shows the start screen. Input can be provided either by cut and paste or through file upload. In both cases the input should be raw text (uploaded files should have .txt extension). The demo will only give complete output in browsers that are HTML-5 compatible and that allow Javascript. For practical reasons, cut and paste input is limited to 4000 characters and file upload to 300 sentences.

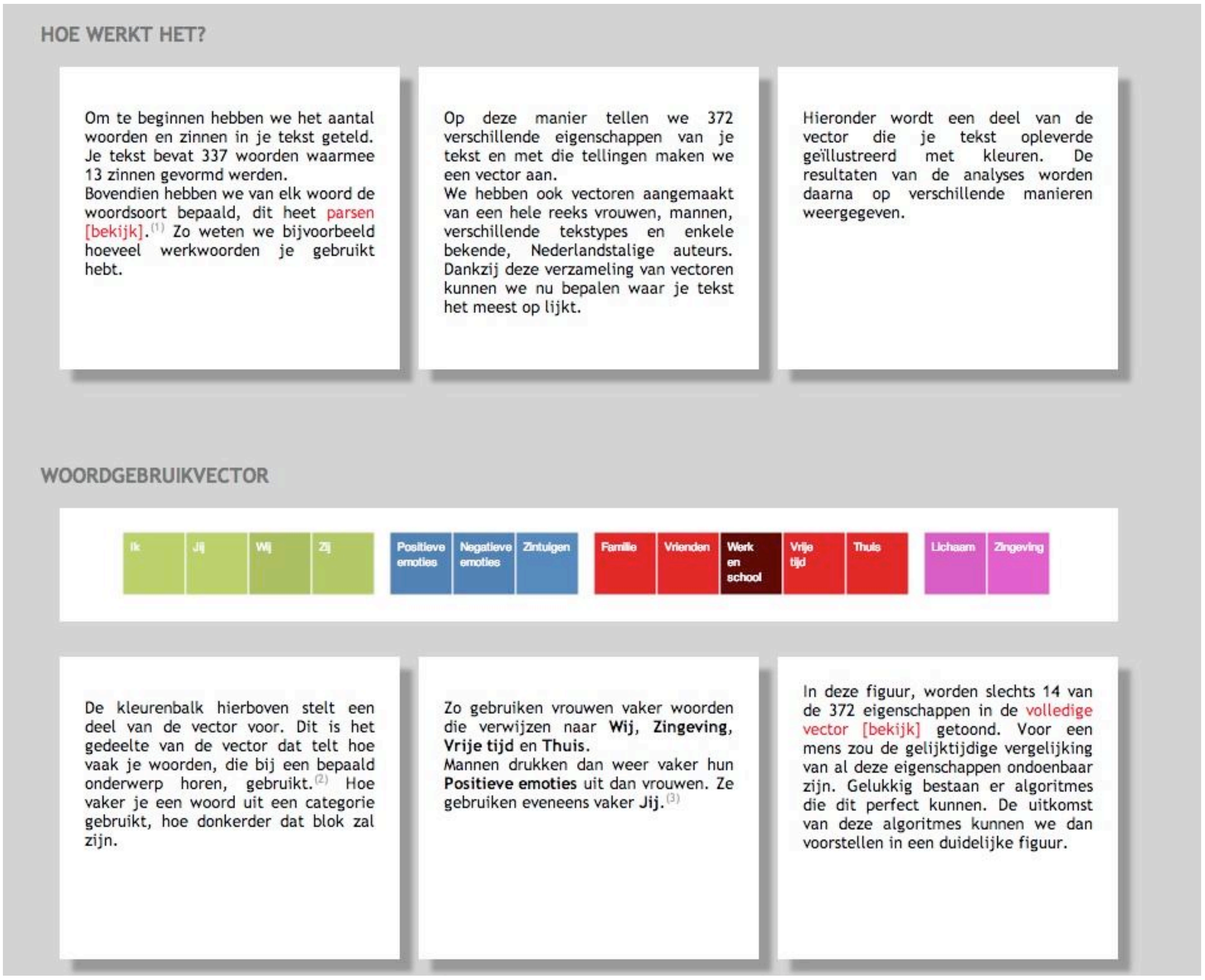

Universiteit Antwerpen
COMPUTATIONAL LINGUISTICS & PSYCHOLINGUISTICS RESEARCH CENTER CLiPS
STYLENE
Analyseer je schrijfstijl
Analyseer
Choose File
No file chosen

Geef hieronder een Nederlandse tekst in waarvan de schrijfstijl geanalyseerd moet worden. Je mag nog 4000 lettertekens ingeven.

Als alternatief kan je een Nederlandse tekst in de vorm van een txt-file, met een maximum van 300 zinnen, uploaden.

Figure 1. Start screen of the stylometry popularization interface.

After entering a text and clicking on ‘analyze’, the software returns a screen with didactic information about the general approach taken in stylometry, and information about different stylometric aspects of the text provided. In Figure 2, introductory information is given about the analysis (users can click through to sample linguistic analyses of the data) and a visual representation of the distribution of words in the text over part of the features used in the system (the full feature representation can be clicked on as well). Darker features represent more frequent features. For the linguistic analysis, Frog was used (Van den Bosch et al, 2007). As features, token unigrams were used and the LIWC features (Pennebaker and Francis, 1996; Pennebaker et al, 2001; Pennebaker et al, 2007).


HOE WERKT HET?

Om te beginnen hebben we het aantal woorden en zinnen in je tekst geteld. Je tekst bevat 337 woorden waarmee 13 zinnen gevormd werden. Bovendien hebben we van elk woord de woordsoort bepaald, dit heet parsen [bekijk].(1) Zo weten we bijvoorbeeld hoeveel werkwoorden je gebruikt hebt.

Op deze manier tellen we 372 verschillende eigenschappen van je tekst en met die tellingen maken we een vector aan. We hebben ook vectoren aangemaakt van een hele reeks vrouwen, mannen, verschillende tekstypes en enkele bekende, Nederlandstalige auteurs. Dankzij deze verzameling van vectoren kunnen we nu bepalen waar je tekst het meest op lijkt.

Hieronder wordt een deel van de vector die je tekst opleverde geïllustreerd met kleuren. De resultaten van de analyses worden daarna op verschillende manieren weergegeven.

WOORDGEBRUIKVECTOR

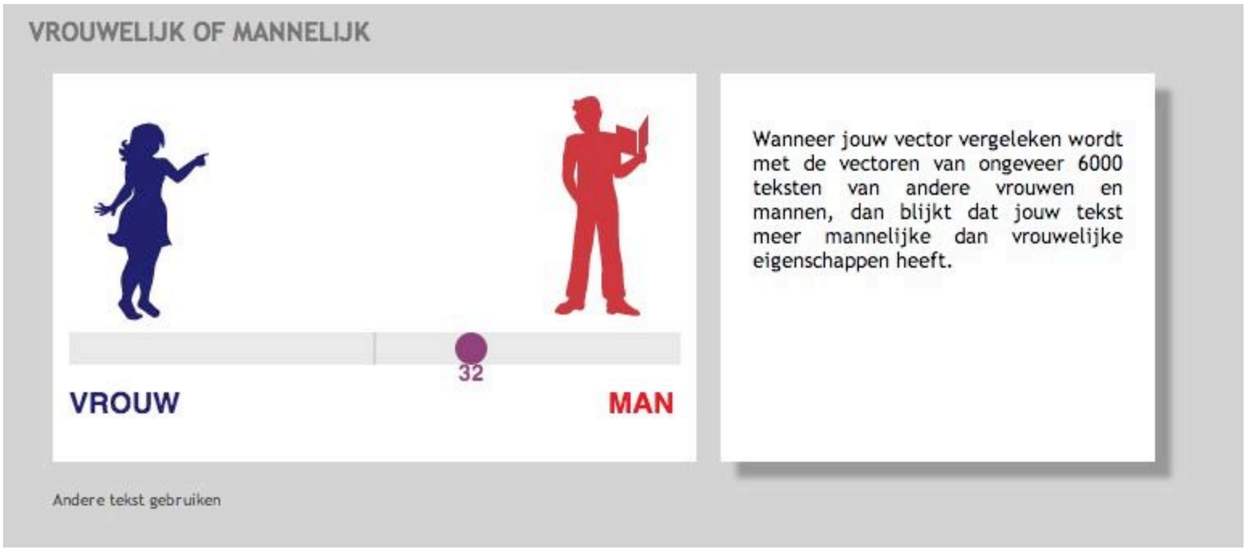
De kleurenbalk hierboven stelt een deel van de vector voor. Dit is het gedeelte van de vector dat telt hoe vaak je woorden, die bij een bepaald onderwerp horen, gebruikt.(2) Hoe vaker je een woord uit een categorie gebruikt, hoe donkerder dat blok zal zijn.

Zo gebruiken vrouwen vaker woorden die verwijzen naar **Wij**, **Zingeving**, **Vrije tijd** en **Thuis**. Mannen drukken dan weer vaker hun **Positieve emoties** uit dan vrouwen. Ze gebruiken eveneens vaker **Jij**.(3)

In deze figuur, worden slechts 14 van de 372 eigenschappen in de volledige vector [bekijk] getoond. Voor een mens zou de gelijktijdige vergelijking van al deze eigenschappen ondoenbaar zijn. Gelukkig bestaan er algoritmes die dit perfect kunnen. De uitkomst van deze algoritmes kunnen we dan voorstellen in een duidelijke figuur.

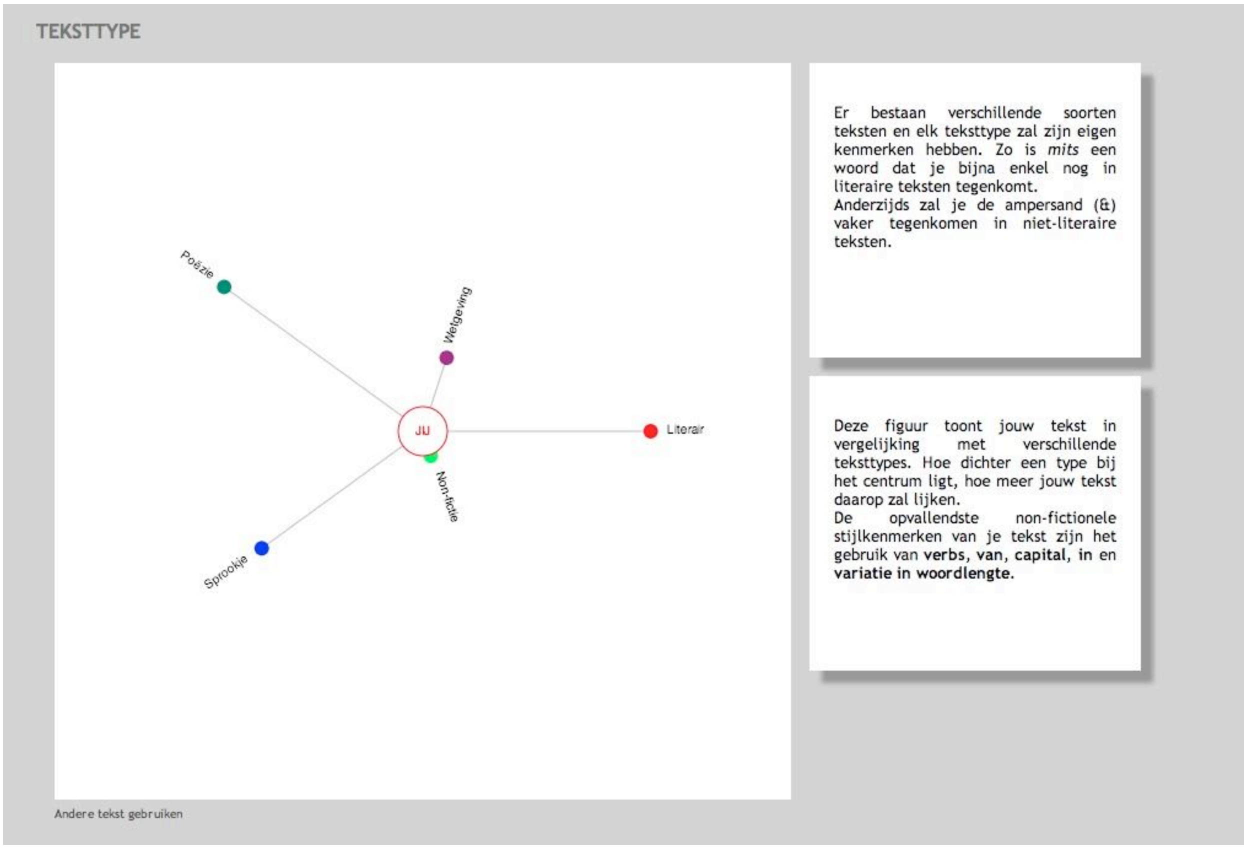
Figure 2. Stylometry popularization interface: output (general).

Figures 3-5 show the additional information that is provided by the demo system: a guess of the gender of the author (based on a simple svm model using all features and trained on part of the CGN data, Corpus Gesproken Nederlands, 2004) (Figure 3); a guess of the register of the text (Figure 4, based on a small corpus that is collected solely for this demo); and a representation of the closeness to samples of the work of a random selection of different Dutch and Flemish authors (Figure 5, based on, on average, the first 11,000 tokens of one of their novels). The gender infobox is the result of assessing the proportion of male and female labels of 71 vectors that are in the vicinity of the vector that is based on the input. Using cosine similarity. The other two infoboxes are based on the Dice coefficient between the different (normalized) vectors.



Wanneer jouw vector vergeleken wordt met de vectoren van ongeveer 6000 teksten van andere vrouwen en mannen, dan blijkt dat jouw tekst meer mannelijke dan vrouwelijke eigenschappen heeft.

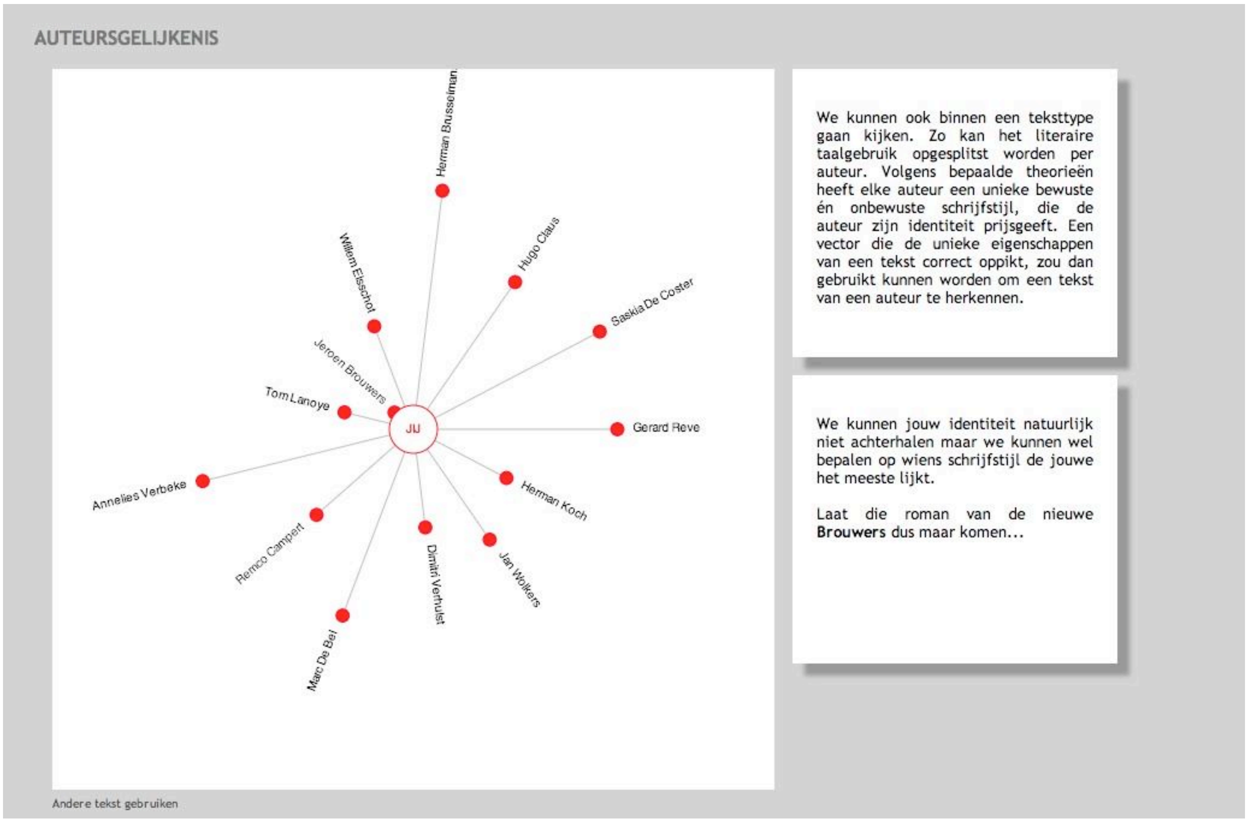
Figure 3. Stylometry popularization interface: output (gender).



Er bestaan verschillende soorten teksten en elk teksttype zal zijn eigen kenmerken hebben. Zo is *mits* een woord dat je bijna enkel nog in literaire teksten tegenkomt.
Anderzijds zal je de ampersand (&) vaker tegenkomen in niet-literaire teksten.

Deze figuur toont jouw tekst in vergelijking met verschillende teksttypes. Hoe dichter een type bij het centrum ligt, hoe meer jouw tekst daarop zal lijken.
De opvallendste non-fictionele stijlkenmerken van je tekst zijn het gebruik van **verbs**, **van**, **capital**, **in** en **variatie in woordlengte**.

Figure 4. Stylometry popularization interface: output (Genre).



We kunnen ook binnen een teksttype gaan kijken. Zo kan het literaire taalgebruik opgesplitst worden per auteur. Volgens bepaalde theorieën heeft elke auteur een unieke bewuste én onbewuste schrijfstijl, die de auteur zijn identiteit prijsgeeft. Een vector die de unieke eigenschappen van een tekst correct oppikt, zou dan gebruikt kunnen worden om een tekst van een auteur te herkennen.

We kunnen jouw identiteit natuurlijk niet achterhalen maar we kunnen wel bepalen op wiens schrijfstijl de jouwe het meeste lijkt.

Laat die roman van de nieuwe **Brouwers** dus maar komen...

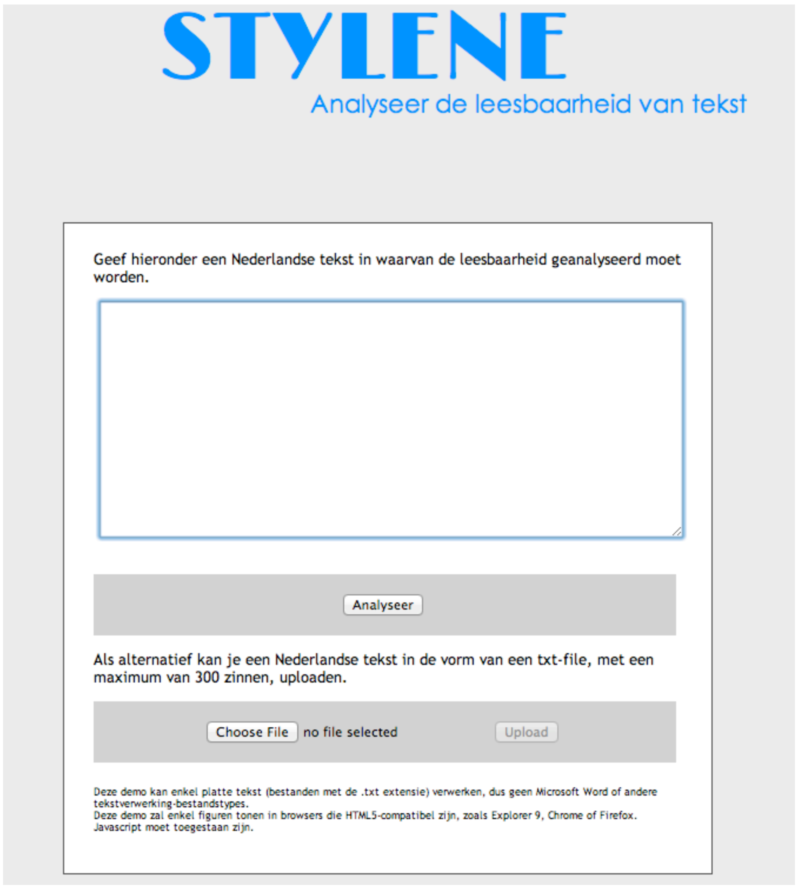
Figure 5. Stylometry popularization interface: output (Distance to authors).

It should be noted that the models used are simplified, and that we don't have any scientific claims about the selection of genres and authors, and about the meaning of the output of the system. The only goal of this interface is to show what computational stylometry is, and provide a feeling for the type of information it uses and the type of output it produces.

2 The Readability Interface

Automatic readability prediction has a long and rich tradition. Research in the 20th century, fueled especially by educational purposes, has resulted in a large number of readability formulas. Typically, these yield either an absolute score (Flesch, 1948; Brouwer, 1963) or a grade level for which a text is deemed appropriate (Dale and Chall, 1948; Gunning, 1952; Kincaid et al., 1975) and are based on shallow text characteristics such as average word and sentence length and word familiarity.

In this readability demo, we present a re-implementation of several readability formulas and propose a new readability method which does not only take into account superficial text characteristics, but also takes into account syntactic text complexity. The software underlying the interface was developed in the framework of a project sponsored by the University College Ghent. Substantial contributions to the readability prediction system were made by Orphée De Clercq, Philip van Oosten, Dries Tanghe, and Veronique Hoste. The interface was developed by Herwig De Smet and Orphée De Clercq. Figure 6 shows the start screen of the demo. Input can be provided either by cut and paste or through file upload. In both cases the input should be raw text. The uploaded files should have the .txt extension.



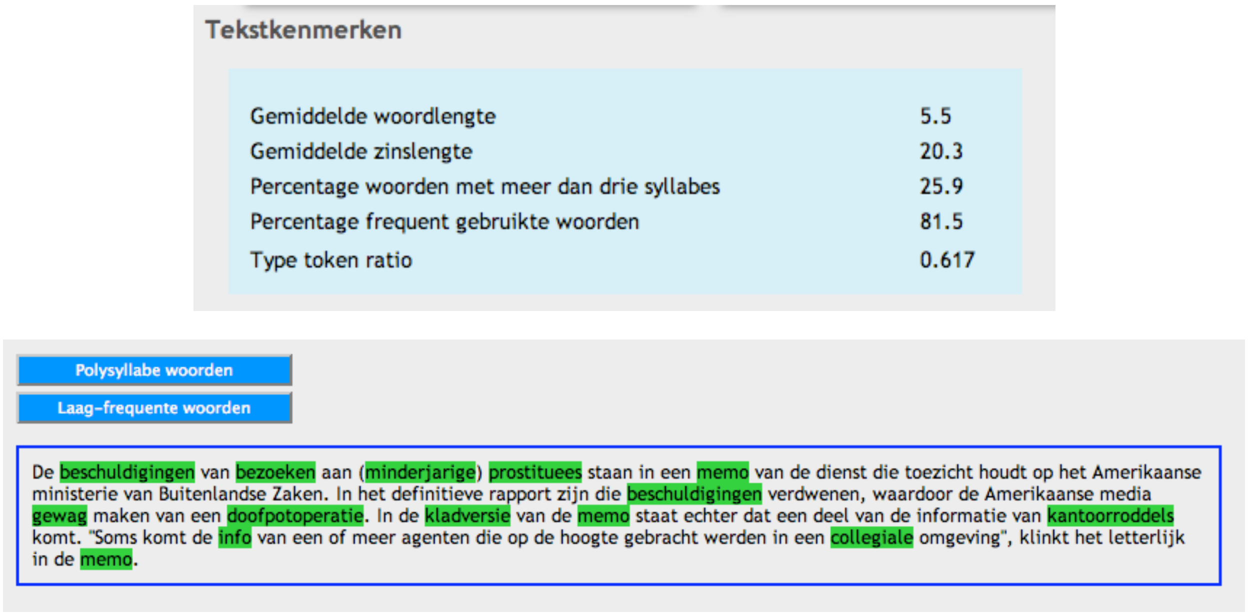
Geef hieronder een Nederlandse tekst in waarvan de leesbaarheid geanalyseerd moet worden.

Als alternatief kan je een Nederlandse tekst in de vorm van een txt-file, met een maximum van 300 zinnen, uploaden.

Deze demo kan enkel platte tekst (bestanden met de .txt extensie) verwerken, dus geen Microsoft Word of andere tekstverwerking-bestandstypes.
Deze demo zal enkel figuren tonen in browsers die HTML5-compatibel zijn, zoals Explorer 9, Chrome of Firefox.
Javascript moet toegestaan zijn.

Figure 6. Start screen of the readability demo

First (Figure 7), the user is presented some general text characteristics of the text he/she entered, such as average word and sentence length, the number of polysyllable words, the percentage of frequently used words and the type token ratio (measure indicating lexical density).



Tekstkenmerken

Gemiddelde woordlengte	5.5
Gemiddelde zinslengte	20.3
Percentage woorden met meer dan drie syllabes	25.9
Percentage frequent gebruikte woorden	81.5
Type token ratio	0.617

De beschuldigingen van bezoeken aan (minderjarige) prostituees staan in een memo van de dienst die toezicht houdt op het Amerikaanse ministerie van Buitenlandse Zaken. In het definitieve rapport zijn die beschuldigingen verdwenen, waardoor de Amerikaanse media gewag maken van een doofpotoperatie. In de kladversie van de memo staat echter dat een deel van de informatie van kantoorroddels komt. "Soms komt de info van een of meer agenten die op de hoogte gebracht werden in een collegiale omgeving", klinkt het letterlijk in de memo.

Figure 7. General text characteristics of the entered text

In a second step, a number of readability formulas are applied to the text which was entered by the user. A readability formula is a mathematical formula intended for indicating the difficulty of text. The formula typically consists of a number of variables, which are characteristics of the text (as displayed in Figure 7), and constant weights. The following text characteristics are used in the formulas displayed in the Readability Interface:

- `avgnumsyl`: Average word length in number of syllables.
- `avgsentencelen`: Average sentence length in number of words.
- `avgwordlen`: Average word length in number of characters.
- `freq77`: Percentage of words also found in a Dutch word list with a cumulative frequency of 77%. The list is based on a list ordered by descending frequency in the "27 Miljoen Woorden Krantencorpus 1995", which is available through the HLT agency at http://tst.inl.nl/en/producten.
- `psw`: Percentage of sentences per word.
- `ttr`: Type/token ratio, the number of unique words divided by the total number of words.
- `freq3000`: Percentage of words not on the Dale-Chall (1948) word list. The Dale-Chall word list contains 3000 of the most frequent words in the English language.
- `avgpolysylsent`: Average number of words of 3 or more syllables per sentence.
- `ppolysylword`: Percentage of words of three or more syllables.
- `ratiolongword`: Ratio of words of more than 6 characters.

In our interface, we compute a number of readability formulas which are designed for either Dutch (Douma, 1960; Brouwer, 1963; Staphorsius, 1994), English (Dale and Chall, 1948; Flesch, 1948; Gunning, 1952; Senter and Smith, 1967; McLaughlin, 1969; Coleman and Liau, 1975; Kincaid et al., 1975) or Swedish (Björnsson, 1968). As it has been shown by van Oosten et al. (2010), that there is a strong correspondence between the readability formulas intended for different languages, all readability formulas are displayed in the interface independently of the language they aim to model.

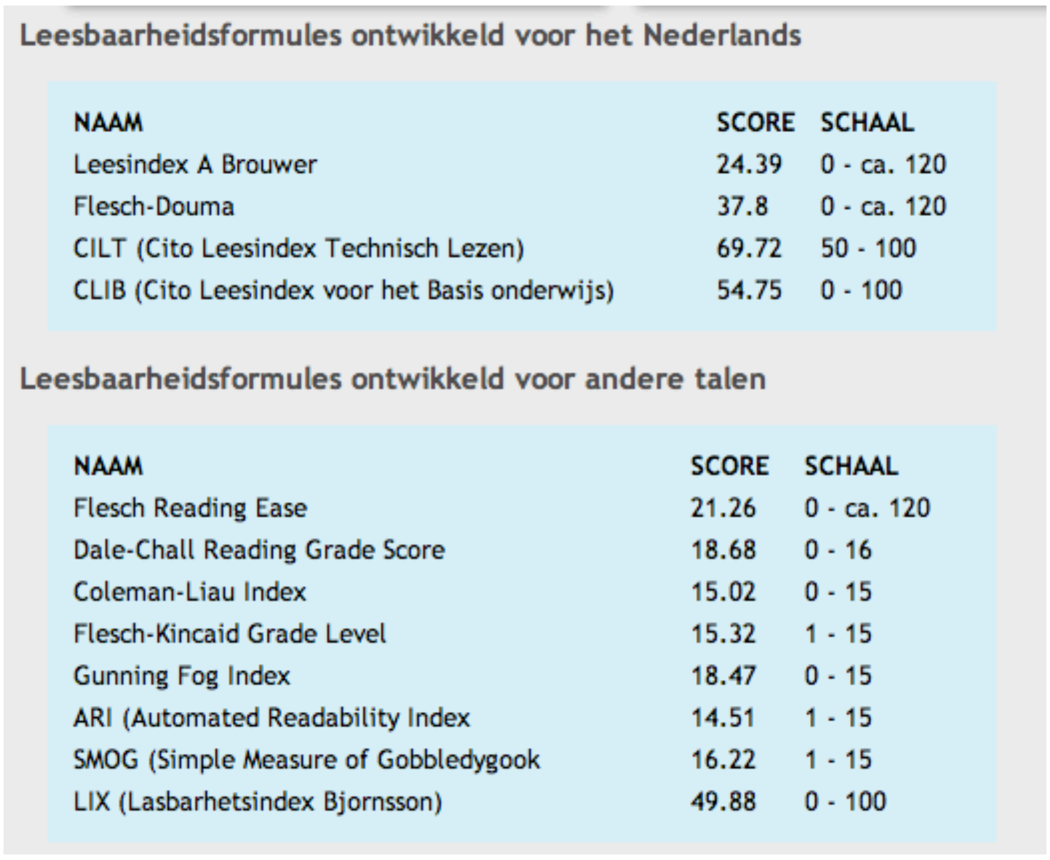
The following readability formulas are displayed in the interface:

For Dutch:
- Leesindex Brouwer (`195 - 2 x avgsentencelen - 67 x avgnumsyl`)
- Flesch-Douma (`207 - 0.93 x avgsentencelen - 77 x avgnumsyl`)
- CILT: Cito leesindex technisch lezen (`114 + 0.28 x freq77 - 12 x avgwordlen`)
- CLIB: Cito leesbaarheidsindex voor het basisonderwijs (`46 +  0.47 x freq77 - 6.6 x avgwordlen - 0.37 x ttr + 1.4 x psw`)

For English:
- Flesch Reading Ease (`207 - avgsentencelen - 85 x avgnumsyl`)
- Dale-Chall Reading Grade Score (`0.16 x freq3000 + 0.05 x avgsentencelen + 3.6`)
- Coleman-Liau Index (`5.9 x avgwordlen - 0.3 x avgsentencelen - 16`)
- Flesch-Kincaid Grade Level (`0.39 x avgsentencelen + 12 x avgnumsyl - 16`)
- Gunning Fog Index (`0.4 x (avgsentencelen + ppolysylword)`)
- ARI: Automated Readability Index (`4.7 x avgwordlen + 0.5 x avgsentencelen - 21`)
- SMOG: Simple Measure of Gobbledygook: `√(30 x avgpolysylsent) + 3.1`

For Swedish:
- Läsbarhetsindex Björnsson: `avgsentencelen + ratiolongword`

Figure 8 displays the results of the different formulas for a text which was entered by the user. The last column gives information on the scale on which the formulas are calculated. For some formulas (all English formulas except for Flesch Reading Ease, the Swedish formula and the Dutch CLIB and CILT), a higher score applies to a more difficult text and a lower score to a more readable text. Their slope is considered positive. For the other formulas, viz. Flesch-Reading Ease, Flesch-Douma and Leesindex Brouwer, the situation is exactly opposite and the slope is considered negative.



Leesbaarheidsformules ontwikkeld voor het Nederlands

NAAM	SCORE	SCHAAL
Leesindex A Brouwer	24.39	0 - ca. 120
Flesch-Douma	37.8	0 - ca. 120
CILT (Cito Leesindex Technisch Lezen)	69.72	50 - 100
CLIB (Cito Leesindex voor het Basis onderwijs)	54.75	0 - 100

Leesbaarheidsformules ontwikkeld voor andere talen

NAAM	SCORE	SCHAAL
Flesch Reading Ease	21.26	0 - ca. 120
Dale-Chall Reading Grade Score	18.68	0 - 16
Coleman-Liau Index	15.02	0 - 15
Flesch-Kincaid Grade Level	15.32	1 - 15
Gunning Fog Index	18.47	0 - 15
ARI (Automated Readability Index	14.51	1 - 15
SMOG (Simple Measure of Gobbledygook	16.22	1 - 15
LIX (Lasbarhetsindex Bjornsson)	49.88	0 - 100

Figure 8. readability scores for entered text

As an alternative to the readability formulas which only base their readability judgement on shallow text characteristics such as word and sentence length, we designed a corpus-based readability prediction system. In order to compile the gold standard underlying the system, two web applications were designed to collect readability assessments for Dutch and English texts: one that is intended exclusively for language experts and one that is open to the general public. Both applications are available under the following link: http://lt3.hogent.be/tools/.

Figure 9 gives an overview of these text scores assigned to texts by the experts and the crowd. The red line in both figures shows how our corpus-based readability prediction system, called Hendi, scores the text compared to the other texts in the corpus. More information on the Hendi system is given in De Clercq et al. (2013).

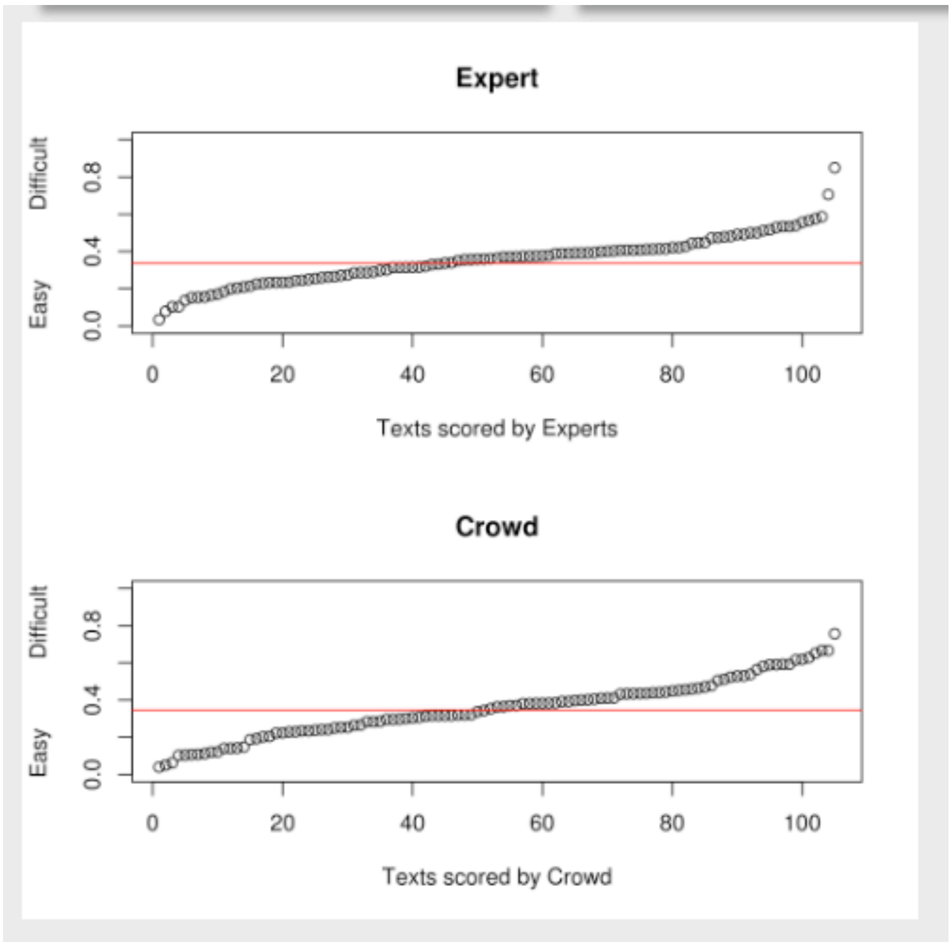
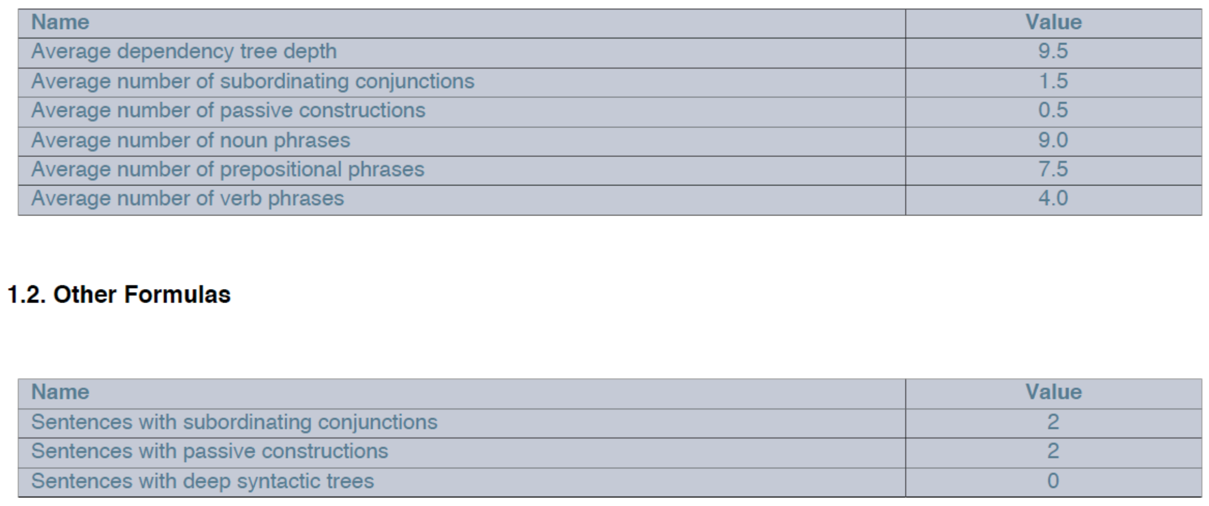


Figure 9: Hendi readability score of the text under consideration in comparison to the expert and crowd readability assessments of all texts in the training corpus

Two flavors of the Hendi system are integrated in this interface: a system which mainly relies on lexical text characteristics, and a second system which also integrates information on the syntactic complexity of the text. In the latter system, the syntactic information as displayed in Figure 10 is taken into account. As the parsing of the text may take some time, this calculation is performed offline and a pdf report is sent to the user as soon as the text is fully processed.



Name	Value
Average dependency tree depth	9.5
Average number of subordinating conjunctions	1.5
Average number of passive constructions	0.5
Average number of noun phrases	9.0
Average number of prepositional phrases	7.5
Average number of verb phrases	4.0

1.2. Other Formulas

Name	Value
Sentences with subordinating conjunctions	2
Sentences with passive constructions	2
Sentences with deep syntactic trees	0

Figure 10. Syntactic information calculated on the basis of the dependency tree of the sentence under consideration

3 The Stylometry Learning Interface

De Stylometry Machine Learning (ML) interface makes possible experiments following the full text categorization approach to stylometry: it allows the linguistic analysis of Dutch language documents, the extraction of features used regularly in the research literature, creation of instances for ML experiments using these features, and the ML experiments themselves. The functionality of the underlying stand-alone system on which this interface is based was designed by Walter Daelemans and implemented by Herwig De Smet. We will describe here the different steps to use it in turn. The interface itself contains helpful hints, examples, and information as well. The system available through the interface has reduced functionality compared to the full stand-alone system that is described in the appendix.

First provide your email address in the appropriate field of the interface so that results can be sent to that address. Then apply the following procedure.

STEP 1. PREPARING AND UPLOADING YOUR DATA FOR TRAINING

The goal of a supervised ML experiment is to use examples of some mapping to learn a model that generalizes to other similar data. For example, on the basis of a number of texts we know to have been written by Willem Elsschot and other texts written by other authors, we train a machine learning method to learn a model of the style of Elsschot. Afterwards we can test the accuracy of this model by applying it to texts that we did not use for training. The interface therefore makes a distinction between a Training run and a Testing run. You start with uploading data for training.

Suppose we want to do a stylometry experiment predicting the gender of the author of tweets. We create a directory with two subdirectories (one for male, one for female), and put the "train" tweets each in a separate file in their corresponding subdirectory. All files should be .txt files with utf8 encoding. After creating a .zip file by compressing this directory (a directory that has as many subdirectories as classes and with the texts belonging to each class in their corresponding subdirectory), this archive can be uploaded for training. After uploading, a result screen is presented indicating successful uploading and providing an identity number for further use.

STEP 2. DEFINING THE EXPERIMENTAL PARAMETERS

To set up the way the data you uploaded will be treated in building a model about style, several types of information have to be provided. First of all, a name has to be provided for the corpus (i.e. the data) that you have uploaded. E.g., Elsschot-1 or Gender-twitter etc. This could for example be the name of the top directory in which you provided subdirectories with training texts.

Next, up to three "analyses" can be provided. An analysis in this context is a specific definition of the information that will be used to represent the text for the ML algorithm (the so-called document representation or instance definition). To define an analysis, select a type, n-gram size, and frequency counting method. Analysis types supported are token (the tokenized words occurring in the text), character (the characters occurring in the text), lemma (the lemmatized tokens in the lext), and pos (the part of speech, or grammatical category, of the words in the text). The n-gram size refers to the length of the sequences that we take into account. E.g. for characters, n set to 3 would select all the character trigrams occurring in the text. A sentence such as *"Give me a break!"* would result in the following character trigrams: *"=Gi, Giv, iv=, =me, me=, =a=, =br, bre, rea, eak, ak=, =!="*.

Analogously, when selecting n = 2 with tokens would result for the same sentence in the token bigrams *"= Give, Give me, me a, a break, break !, ! ="*. Additional information to be provided for each analysis is the frequency count type which can be absolute (how many times does a particular feature, for example the character trigram "=!=" occur in the document) or relative (what is the proportion of the occurrences of this feature in all the occurrences of all features in the document).

For each analysis specified, two datasets will be generated: one where document representations consist of binary vectors, and one where they consist of numeric vectors (where the numeric values are absolute or relative as selected by the user). In addition a dictionary is provided with the selected features for that experiment, their position in the document vector, and their frequency.

STEP 3. SELECTING THE FEATURES

The document representation defined in the previous step can be very large. In the "filter" step, this set of features can be reduced to a manageable number on the basis of frequency, informativeness or a combination of both.

There are three filters that can optionally be selected. If none is selected, all features will be used. The total set filter allows defining a frequency band. For example, we might be interested in selecting the 10% most frequent features (set upper percentage to ten and leave lower percentage at 0), the 50% least frequent features (set upper percentage to 0 and lower percentage to 50), or the middle band (in case you want the features that are neither very frequent or infrequent). In that case both thresholds could be set to 20, for example. Remember that "features" refers in this context to the items generated for the document representation, such as character trigrams or lemma bigrams.

Not all features are equally relevant for distinguishing between classes. Statistical and information-theoretic methods such as chi-squared and information entropy can be used to analyze the degree in which a particular feature (e.g. the character trigram '=!=') can differentiate between the classes. The two remaining filters order the features according to relevance as defined by these methods and allow the selection of a percentage of these most relevant features.

All that remains to be done now at this stage is indicating whether you want document features to be computed (average word length, average sentence length, average number of syllables, number of hapax legomena, number of hapax dis legomena and readability), which Machine Learning algorithm you want to use, and which document representation you want to use (binary or numeric). By clicking start, the whole process will be activated and an id generated.

By email you will receive a zip file that contains all instance vectors for the analyses and filters you have chosen for the current training run. The mail also contains a unique identifier that is used as a link between the training run and a test run you may want to perform in relation to this training run.

STEP 4. TESTING

With the identifier provided, you can enter the stylene machine learning interface again, this time with a test data set submitted in the same format as the training data. The trained model will be applied on the test data provided and an analysis will be returned.

By email you will receive a zipfile that contains all the instance vectors that have been generated for this testrun.

3.1 Using the interface for text analysis only (optional)

In case you are interested only in parsing your text(s), it is possible to go to the Frog parser interface, and submit an archive of texts (again following a zip archive format now with one directory of files to be analysed) that will then only be parsed. No ML models will be built in that case and with each input file in the archive, a Frog output file will appear with the parsed input text. The Frog parser is also accessible from the Readability interface. The Frog parser used for this project is frozen at version 0.12.15 (c) ILK 1998 - 2012 to prevent compatibility issues in the future.

4 Conclusion

The Stylene project, funded by EWI and executed by the CLiPS and LT3 research groups, resulted in a number of resources, assembled behind a single interface, that we hope will prove useful for different categories of users. People interested in the computational linguistics applications of stylometry and readability can analyze texts and be educated about the types of analysis that these research fields apply. Users in the digital humanities can test the automatic text categorization approach to stylometry in a user friendly interface suited for exploratory research. Whereas the first stylometry interface is based on simplified models, the readability interface and the machine learning of stylometry interfaces fall back on state of the art software for Dutch. In addition, the interface provides an easy access to the state of the art Dutch text analysis software package Frog.

5 References

Björnsson, C-H. (1968). Läsbarhet. Almqvist and Wiksell, Stockholm.

Brouwer, R. H. M. (1963). Onderzoek naar de leesmoeilijkheden van Nederlands proza. Pedagogische Studiën, 40:454–464.

Coleman, M. and Liau, T. L. (1975). A computer readability formula designed for machine scoring. Journal of Applied Psychology, 60:283–284.

Corpus Gesproken Nederlands (CGN) (2004). Nederlandse Taalunie. http://tst-centrale.org/images/stories/producten/documentatie/cgn_website/doc_Dutch/start.htm (Last accessed: June 2013).

Daelemans, W. (2013). Explanation in computational stylometry. Computational linguistics and intelligent text processing: 14th International Conference, CICling 2013, Samos, Greece, March 24-30, 451-462.

Dale, E. and Chall, J. S. (1948). A formula for predicting readability. Educational research bulletin, 27:11–20.

De Clercq, O., Hoste, V., Desmet, B., van Oosten, P., De Cock, M., & Macken, L. (2013). Using the Crowd for Readability Prediction. Natural Language Engineering, 1-33. Cambridge Journals Online.

Douma, W. (1960). De leesbaarheid van landbouwbladen: een onderzoek naar en een toepassing van leesbaarheidsformules. Bulletin, 17.

Flesch, R. (1948). A new readability yardstick. Journal of Applied Psychology, 32(3):221–233.

Gunning, R. (1952). The technique of clear writing. McGraw-Hill, New York.

Kincaid, J. P., Jr., R. P. F., Rogers, R. L., and Chissom., B. S. (1975). Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel. Research branch report RBR-8-75, Naval Technical Training Command Millington Tenn Research Branch, Springfield, Virginia.

McLaughlin, G.H. (1969). SMOG grading - a new readability formula. Journal of Reading, pp. 639–646.

Pennebaker, J. W. and Francis, M.E. (1996). Cognitive, emotional, and language processes in disclosure. Cognition and Emotion 10 (6), pp. 601-626.

Pennebaker, J. W., Francis M.E., and Booth R.J. (2001). Linguistic Inquiry and Word Count (LIWC): LIWC2001. Mahwah: Lawrence Erlbaum Associates.

Pennebaker, J. W., Francis M.E., and Booth R.J. (2007). Linguistic Inquiry and Word Count (LIWC): LIWC2007. http://www.liwc.net (Last accessed: June 2013).

Senter, R. J. and Smith, E. A. (1967). Automated readability index. Technical Report AMRLTR-66-220, University of Cincinnati, Cincinnati, Ohio.

Staphorsius, G. (1994). Leesbaarheid en leesvaardigheid. De ontwikkeling van een domeingericht meetinstrument. Cito, Arnhem.

Staphorsius, G. and Krom, R. S. (1985). Cito leesbaarheidsindex voor het basisonderwijs: verslag van een leesbaarheidsonderzoek. Number 36 in Specialistisch bulletin. Cito Arnhem.

Van den Bosch, A., Busser, G.J., Daelemans, W., and Canisius, S. (2007). An efficient memory-based morphosyntactic tagger and parser for Dutch, In F. van Eynde, P. Dirix, I. Schuurman, and V. Vandeghinste (Eds.), Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting, Leuven, Belgium, pp. 99-114.

Van Halteren, H., Baayen, R.H., Tweedie, F., Haverkort, M. and Neijt, A. (2005). New Machine Learning Methods Demonstrate the Existence of a Human Stylome. In Proceedings of Journal of Quantitative Linguistics. pp. 65-77.

Van Oosten, P., Tanghe, D., and Hoste, V. (2010). Towards an Improved Methodology for Automated Readability Prediction. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., and Tapias, D., editors, Proceedings of the seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta. European Language Resources Association.

Appendix I. Manual Stand-Alone System

The Stylene workflowmanager is a Java application which processes a corpus of documents that belong to specific classes, and generates instance vectors from them that can be used as training and testing data for Machine Learning.This manual describes the important parts of the application and how to use them and adjust the parameters that manage the procedure.

How to run the program?

To run the program, open a command line window, go to the folder where `stylene.jar` is located and type the following command:

```
      java -jar stylene.jar
```

The executable jar first looks for a configuration file (`configuration.xml`). If this file is in the same folder as the jar, the jar will find it. If not, you have to specify the path to the file as an argument.

```
      java -jar stylene.jar /home/projects/stylene/configuration.xml
```

The following 4 files have to be present in the same folder as the jar:

`startparameters.xml`	parameters that manage the process
`startparameters.xsd`	validation of startparameters.xml
`logback.xml`	default path for Logback for Java
`styleneruns.xml`	keeps track of all the workflow runs

WHAT IS IN CONFIGURATION.XML?

The entries in this file contain the paths to the components the application needs.

```
<properties>
       <entry key="main-path">/home/nlpsoftware/stylenerun/</entry>
       <entry key="start-parameters path">/home/nlpsoftware/stylenerun/startparameters.xml</entry>
       <entry key="start-parameters-xschema-path">/home/nlpsoftware/stylenerun/startparameters.xsd</entry>
       <entry key="logback-path-standalone">/home/nlpsoftware/stylenerun/</entry>
       <entry key="log-path">/home/nlpsoftware/stylenerun/log/stylenerun.log</entry>
       <entry key="data-path">/home/nlpsoftware/stylenerun/data/</entry>
       <entry key="xsl-path">/home/nlpsoftware/stylenerun/xsl/</entry>
       <entry key="frog-path">/usr/bin/</entry>
       <entry key="svm-path">/usr/bin/</entry>
       <entry key="timbl-path">/usr/bin/</entry>
       <entry key="weka-path">/usr/bin/</entry>
</properties>
```

- `main-path`: the main path of the application.
- `start-parameters-path`: the path to the file `startparameters.xml` (see below).
- `start-parameters-xschema-path`: the path to the file `startparameters.xsd`, which is used for validation of the startparameters (see below).
- `logback-path-standalone`: the path to the file `logback.xml` that contains the parameters for the logback framework.

- `log-path`: the path to the log file.
- `data-path`: the folder where the data has to be copied to (below).
- `xsl-path`: the folder that contains the xsl code.
- `frog-path`: the path to an installed version of Frog.
- `svm-path`: the path to an installed version of svm.
- `timbl-path`: the path to an installed version of timbl.
- `weka-path`: the path to an installed version of weka.

What is in startparameters.xml ?

In this file you have to specify the values of all the parameters which will guide the workflow, before actually starting the run. The language used is xml. When the application starts, it first reads the entries in this file and validates them to the XSchema defined in `startparameters.xsd`. If something goes wrong in this validation step, the process cannot start. The error is logged and shown to the user.

Below is a list of all the entries in the file, together with all possible values.

```
<run-type>train</run-type>
```

Before you start the application, you have to split up your corpus in a training and a test set. In a Stylene workflow, you first let the application generate instance vectors for the training set which can then be processed by a machine learner. In the test phase, the same procedure is followed, but now for the test set. In the `run-type` element you specify whether this is a training run or a test run.

Possible values: `test` | `train`

```
<UUID-training-run>fd615280-9ceb-4de0-aaab-2302bac680b4</UUID-training-run>
```

For each workflow run, an entry is made in `styleneruns.xml`. For a training run, a unique id is saved in the element `UUID` of the entry. This id will be the link between a training and a test run.

If you are doing a training run, `UUID-training-run` is ignored. For a test run, you have to copy the UUID of the corresponding training run into `UUID-training-run`.

```
<run-number>1</run-number>
```

The number of the run for this corpus. This allows for multiple runs on the same corpus, but with different parameters.

```
<corpus-name>personae corpus</corpus-name>
```

The name of the corpus.

```
<number-of-classes>3</number-of-classes>
```

As will be described below, you have to organize your documents into classes. Each class corresponds to a folder on the hard drive. Here you specify the number of classes.

```
<parsers>
    <parser>
        <name>frog</name>
    </parser>
</parsers>
```

If your documents are plain text files, they first have to be parsed. In the current version, only the Frog parser is supported. If you want parsing to be applied, use this element. If not, leave it out.

Possible values: `frog`

```
<parse-table>
    <number-of-columns>10</number-of-columns>
    <separation-character>tab</separation-character>
</parse-table>
```

This entry contains information about the structure of the parsed documents. In the current version of Frog, the number of columns is 10, and each entry is separated by a tab character.

```
<analyses>
    <analysis>
        <name>token</name>
        <n>2</n>
        <column>1</column>
        <count-type>absolute</count-type>
    </analysis>
    <analysis>
        <name>character</name>
        <n>2</n>
        <column>1</column>
        <type>sentence</type>
        <count-type>relative</count-type>
    </analysis>
    <analysis>
        <name>lemma</name>
        <column>1</column>
        <count-type>absolute</count-type>
    </analysis>
    <analysis>
        <name>pos</name>
        <n>2</n>
        <column>1</column>
        <count-type>absolute</count-type>
    </analysis>
    <analysis>
        <name>skipgram</name>
        <n>2</n>
        <skip-n>3</skip-n>
        <column>1</column>
        <count-type>absolute</count-type>
    </analysis>
</analyses>
```

Here you specify which analyses you want to apply to your documents. For each analysis element, the following values have to be specified:

`name` of the analysis:

`token` | `lemma` | `pos` | `character` | `skipgram` `n` value:

The level of n-gramming

`column` in the parsetable:

The column in the parsetable that contains the element to be analysed

`count-type`: absolute frequencies or relative (normalized) frequencies:

`relative` | `absolute`

For the character analysis, you have to specify the level at which n-gramming will be applied:

`word` | `sentence` | `document`

For the skip-gram analysis, you have to specify the number of the element which has to be skipped.

```
<meta-analyses>
    <meta-analysis>
        <name>Text Statistics</name>
        <column>2</column>
    </meta-analysis>
</meta-analyses>
```

In the current version, you can specify only 1 type of meta-analysis, the text statistics or document features. If you want this analysis to be applied you have to enter the element. If not, leave this element out.

```
<filters>
    <filter>
        <name>total set</name>
        <range>
            <!-- Either index or percentage, not both ! -->
            <index>
                <upper>0</upper>
                <lower>100</lower>
            </index>
            <percentage>
                <upper>100</upper>
                <lower>98</lower>
            </percentage>
        </range>
    </filter>
    <filter>
         <name>chi square</name>
         <cut-off>100</cut-off>
         <validity-cut-off>2</validity-cut-off>
    </filter>
    <filter>
        <name>information gain</name>
        <cut-off>25</cut-off>
        <validity-cut-off>2</validity-cut-off>
        <gain-weighted status="on" weight="0.8"/>
    </filter>
</filters>
```

The `filters` element determines which n-grams will be used to generate the instance vector files. If you don't want to apply filtering, leave this element out.

The total set of n-grams is always the starting point. This list is sorted on frequency. For each specified filter a separate instance vector file is generated.

Possible values:

`total set | chi square | information gain`

TOTAL SET

You can give a range of n-grams that have to be used in 2 ways. If you use `index`, you give an `upper` and a `lower` limit that corresponds to the position in the total set. If you use `percentage` you do the same, but in percentages.

CHI SQUARE

The chi square values are computed on the total set. The result is a list of n-grams, sorted on descending chi square values. The `cut-off` value determines which top range of n-grams you want to use for the instance vectors. The `validity-cut-off` determines how many times an n-gram has to occur in a class to apply for inclusion in the chi square list.

INFORMATION GAIN

The information gain values are computed on the total set. The result is a list of n-grams, sorted on descending information gain values. The `cut-off` value determines which top range of n-grams you want to use for the instance vectors. The `validity-cut-off` determines how many times an n-gram has to occur in a class to apply for inclusion in the information gain list. In the `gain-weighted` element you can specify if you want to weight the information gain values by the frequency of the n-gram, and by how much.

```
<engines>
    <engine>
        <name>svm</name>
        <run>off</run>
    </engine>
    <engine>
        <name>timbl</name>
        <run>off</run>
    </engine>
    <engine>
        <name>weka</name>
        <run>off</run>
    </engine>
</engines>
```

In the `engines` element you can specify for which machine learning engines you want instance vectors to be generated. 3 formats are supported: SVM, Timbl and Weka.

Possible values:

`svm | timbl | weka`

In the name element you write the name of the engine, and in the run element whether you want to run the engine after the instance vectors have been generated.

```
<pruning>
    <frequency>0</frequency>
    <percentage>0</percentage>
</pruning>
```

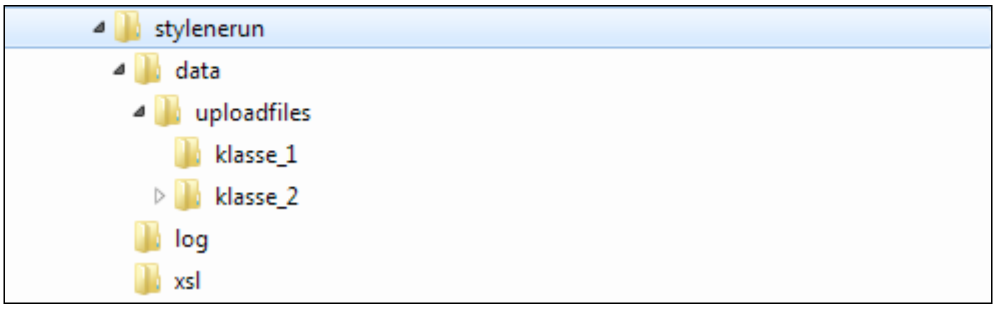
During the analyses you can apply `pruning` to the results. If you use this optional element, the number of n-grams that you specify will not be taken into account. In the case of `frequency`, all n-grams at total set level with a frequency smaller or equal to the value will be pruned. In the case of `percentage` the amount of n-grams with the smallest frequency that corresponds to the value will be pruned. Frequency overrules percentage.

```
 <precision digits="5"/>
```

The `precision` element determines the decimal places used in the calculations.

Folder organization

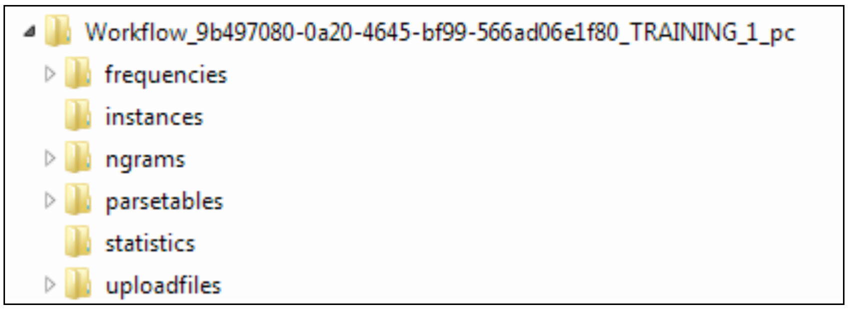
The Stylene workflow manager expects a specific structure of the folders on the hard drive, as shown below:



```
stylenerun
  data
    uploadfiles
      klasse_1
      klasse_2
  log
  xsl
```

In the folder `uploadfiles` you have to copy your documents, for each class a folder.

Each run gets its own temporary folder. The name of this folder starts with `Workflow`, followed by a unique id, the runtype and the name of the corpus. The steps that are followed by the application are reflected in the names of the subfolders and the files.



```
Workflow_9b497080-0a20-4645-bf99-566ad06e1f80_TRAINING_1_pc
  frequencies
  instances
  ngrams
  parsetables
  statistics
  uploadfiles
```

`uploadfiles` contains a copy of the classes and the documents.

In `parsetables` the parser will put its results. If your files have already been parsed, they will be copied to this folder.

`ngrams` contains the n-grams, for each analysis and for all documents.

`frequencies` contains the frequencies of the n-grams, for each analysis.

`statistics` contains the results of the chi square and information gain computations.

`instances` contains the generated instance vectors and the dictionaries.

All files have a meaningful name.

Document features

If the meta-analysis has been executed, instance vectors of the document features will be generated and saved in the instances folder, with corresponding dictionaries.

7 document features are calculated:

- `AverageSentenceLength`: the average number of words per sentence in the document.
- `AverageWordLength`: the average number of characters per word in the document.
- `AverageNumberOfSyllables`: the average number of syllables per word in the document.
- `NumberOfHapax1`: the number of words that occurs 1 time in the document.
- `NumberOfHapax2`: the number of words that occurs 2 times in the document.
- `ReadabilityScore`:

```
206.835 - (1.015 * AverageSentenceLength) - (84.6 * AverageNumberOfSyllables)
```

- `TypeTokenRatio`: the number of types / the number of tokens.

Logging

All important steps in the process are logged to the console and to a logfile. The path to the logback configuration file and to the logfile itself are in the file `configuration.xml`. Each log entry contains a timestamp and a statement:

```
21 Oct 2012 -- 09:05:16.444 - INSTANCE VECTORS for N_GRAM_TOKEN_1
```