Physics Engineering
Bachelor's Thesis

DEVELOPMENT AND EVALUATION OF A SEMI-AUTOMATIC ANONYMIZATION TOOL FOR TEXTUAL DATA

Author: Hèlia Brull Corretger
Director: Jordi Forné
Co-director: José Antonio Estrada

UNIVERSITAT POLITÈCNICA DE CATALUNYA

June 2019

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Abstract

Despite its undeniable advantages, the exponential growth of data analytic capabilities implies a significant increase in the risk of personal privacy loss and re-identification of the individuals appearing in databases. A tool capable of treating the data in a way that would avoid re-identification of the user and, thus, could be shared to other companies or to study groups would be undoubtedly effective.

The main objective of this project focuses on anonymization of textual data. That is, the goal is to create a tool that enables the anonymization of an input text to prevent identity disclosure while conserving as much utility as possible. This is achieved with the help of Information Theory and Natural Language Processing techniques. Additionally, the user interaction is required to make certain decisions, having as a consequence a semi-automatic anonymization tool for textual data. Because of that, we have developed a Graphical User Interface. The evaluation of the tool in automatic mode has been carried out in order to check its performance in function of certain parameters.

Acknowledgements

I would like to thank all the people who have helped me during my Bachelor's thesis, either with knowledge or moral support. First of all my director, professor Jordi Forné, for his guidance and encouragement. Also my co-director, José Antonio Estrada, for his advice.

Special thanks to my friends for their support in difficult and stressful moments throughout these months.

To the ones who have encouraged me and believed in me the most, not only during the realisation of this thesis but throughout the last four years. Pare, Mare, Marcel, Vicenç: moltíssimes gràcies.

List of figures

List of tables

Table of contents

1. Introduction

In this section, the motivation and objectives of this project are presented, as well as the state of the art.

1.1. Motivation and objectives

In the so-call big data era, massive amounts of information are generated every day. The existence and usage of such data has prompted a breakthrough in countless fields, including, among many others, business and marketing while achieving better propositions to customers or healthcare as a result of research improvement and prediction of future events.

Despite its undeniable advantages, the exponential growth of data analytic capabilities implies a significant increase in the risk of personal privacy loss and reidentification of the individuals appearing in databases.

In this data-driven world context, the General Data Protection Regulation (GDPR) was approved by the EU Parliament on 14 April 2016 and applied on 25 May 2018, and since then non-compliant companies may face heavy fines [1]. Its main aim consists of reshaping the way in which data is handled across every sector, from healthcare to banking and beyond. This new regulation implies, among others, that it is mandatory for the companies to ask the user for consent in order to collect their data. Moreover, what data is being collected and what it is used for should also be informed of. These measures are taken because the company possesses the sufficient data to identify one individual among all the others.

Hence, a tool capable of treating the data in a way that would avoid reidentification of the user and, thus, could be shared to other companies or to study groups in order to do research would be undoubtedly effective.

The main objective of this project focuses on anonymization of textual data. That is, the goal is to create a tool that enables the anonymization of an input text to prevent identity disclosure while conserving as much utility as possible. The detection of terms that enable reidentification is done with the combination of different methods. Then, these terms are sanitized, that is, distorted in some way so that they do not identify, via two different techniques: blacking out and generalizing. This process is improved by asking for the user interaction to make certain decisions resulting in a semi-automatic anonymization tool for textual data.

The memory is organized as follows: In chapter 2, a basic theoretical background is offered to enable the understanding of the concepts used throughout the project development. In chapter 3, the development of the tool is exposed step by step, and the different problems encountered and the solutions adopted to solve them are explained. In chapter 4, the evaluation of the tool performance in automatic mode is carried out and the results are discussed. In chapter 5, the final version of the tool is presented along

with the functionalities it contains. In chapter 6, conclusions are drawn and further work is commented.

1.2. State of the art

In the so-call big data era, massive amounts of information are generated every day, the existence and usage of which has prompted a breakthrough in countless fields.

Despite its undeniable advantages, the exponential growth of data analytic capabilities implies a significant increase in the risk of personal privacy loss and reidentification of the individuals appearing in databases.

Because privacy risks have prevailed for years now, many different sanitization methods have been proposed to this day. Specifically, efforts have been put in privacy protection of structured data like relational databases. In that field, authors profit from the structure of data to anonymize attributes that are known to be probable identifiers.

Privacy protection in unstructured data (query logs, raw text documents) has been taken less into consideration despite being the usual way in which data is transferred between parties. Traditionally, sanitization of text documents was done manually by qualified reviewers. That made the process more expensive and time-consuming, apart from not scalable when the data volume grew. Thus, the need for automatic text sanitization methods emerged.

One of the first unsupervised approaches was the Scrub system [2], which focuses on the removal of sensitive terms from medical records, being an example of methods that rely on detection patterns for specific data types. The detected terms are then replaced by others of similar type. The main drawback of patterns related to specific areas of knowledge, as well as trained classifiers, is the restriction in applicability of the approach when sanitization needs to be applied to a heterogeneous scenario.

The ERASE system [3] was developed to perform unstructured text documents sanitization automatically. This is done not by searching for predefined patterns but by detecting and removing sensitive elements using a database of entities (persons, products, diseases, etc.) which aims to model public knowledge. Each entity in the database is associated with a set of terms, which constitute the context of the entity, and some entities are considered protected. Thus, ERASE works by finding common terms between the document and the entities' context, and removing them from the document. In this way, no protected entity can be inferred as being mentioned in the document by an adversary. The main disadvantage of removing sensitive terms is the decrease in the document's utility. Moreover, the obscuration may raise awareness of the document's sensitivity.

Another approach proposed [4] relies on external general-purpose knowledge bases, which broadens the applicability of the method. Because it requires no supervision during the sanitization process, it is a more scalable solution and enables its application to environments with large data volumes. The detection of sensitive terms in this case

relies on quantifying how much information each textual term provides. Thus, a term is considered sensitive if it provides more information than what the attacker is assumed to possess. In order to do that, the Information Content (IC) of each textual term is computed. Because the IC of a term is, formally, the inverse of its appearance probability in a corpus, a common approach to determine this probability in the Web corpus is to use the *hit count* provided by web search engines when querying a term. In this way, general terms provide less IC than more concrete ones, and can be considered less sensitive.

Because semantics are the mean to interpret and extract conclusions when analysing textual data, the removal or hiding of sensitive terms should be carried out in a way that data semantics, and, thus, utility, are conserved.

All sanitization methods to that day suffered from a relevant problem, as the authors of the previous technique stated: the evaluation of sensitivity of textual terms was studied independently from each other. That is a risky approach, taking into account that terms in any textual document are semantically related and, therefore, re-identification of sanitized terms can take place because of the presence of related terms in clear form.

Thus, an automatic sanitization method that detects those terms that are semantically correlated to sensitive ones [5] was proposed, being a complement to any sanitization method which detected sensitive terms independently. It works by removing the terms that may cause disclosure of sensitive ones.

An improvement of this method was developed by its authors [6], enabling the quantification of disclosure risk of semantically correlated terms with a sensitive one whether the latter is removed or generalised. Moreover, the terms that may cause disclosure of sensitive ones are generalized rather than removed, which increases the final document's utility significantly.

An application of this approach is found in the field of social networks [7], in which automatic sanitization is applied to textual data published on the users wall to hide sensitive data from unauthorized third parties.

2. Theoretical background

In this section, a theoretical and methodological background is given. First we will introduce data privacy fundamentals, both in structured and unstructured data. Then, we will review the basics of the information theory approach used. After that, concepts of Natural Language Processing will be presented. Finally, we will explain some useful performance model concepts, focusing on binary classification.

2.1. Data privacy fundamentals

Anonymity of a subject is defined as the fact that the subject is not identifiable within a set of subjects [8]. In this definition, being not identifiable means being not distinguishable from the rest of subjects.

The term **attacker** (or adversary) is used to refer to the set of entities working against some protection goal [8]. The objective of adversaries is to increase their knowledge on the themes of interest.

Another important concept related to privacy is the one of disclosure. **Disclosure** takes place when attackers take advantage of the observation of available data to improve their knowledge on some sensitive information about an item of interest [8]. Therefore, disclosure may be defined in terms of the additional sensitive information or knowledge that an attacker is able to acquire by observing the system.

According to [9], disclosure may be studied regarding two different perspectives:

- **Identity disclosure.** It takes place when the adversary can correctly link a respondent to a particular record in the protected dataset, that is, those pieces of information that can re-identify a protected entity or individual.
- **Attribute disclosure.** It takes place when the adversary can learn something new about an attribute of a protected entity, even when relationship between the individual and the data cannot be established. That is, when the observation of data enables the attacker to increase his accuracy on a certain attribute.

Under the identity disclosure perspective, privacy problems appear when data referring to individuals make them unique. However, only a few attributes are enough to make an individual unique. Usually, it is a combination of many attributes not leading to uniqueness what makes an individual unique. This fact leads to the following classification:

- **Identifiers**: terms that unequivocally identify an individual. An example might be a person's full name, the ID number or Social Security number.
- **Quasi-identifiers**: attributes that may not identify an individual *per se,* but may unequivocally disclose identities when they are combined. This is the case of, for example, the combination of last name, place of birth and job.

These two types of attributes, together with confidential attributes (medical conditions, sexual orientation, …) are referred to as *sensitive information*.

It is actually this information the one that is useful for surveys, studies, etc. Current legislations, however, are very strict with respect to individual privacy protection. Thus, there is always an aim of balancing the trade-off between data privacy and data utility. This ranges between perfect protection, where all sensitive data is encrypted or removed while the utility is entirely destroyed, and direct publication of unprotected data, hence conserving the maximum utility but with no privacy protection at all.

Let us now explore data privacy in structured and unstructured data.

2.1.1. Data privacy in structured data

Many privacy models have been proposed for structured data such as databases. They rely on the fact that they are records with univalued attributes, whether these values re numerical or categorical. This fact makes it easy to define, achieve and evaluate privacy guarantees.

Regarding attribute and identity disclosure in databases, four possibilities may occur [8]:

- Neither identity nor attribute disclosure
- Identity disclosure and attribute disclosure
- No identity disclosure but attribute disclosure
- Identity disclosure but not attribute disclosure

Let us consider the three cases where disclosure is produced.

Case A. Identity disclosure and attribute disclosure.

Let us consider an attacker that has the following information: *(Susan, Cambridge, 58)*. If a hospital publishes a database with all the variables in Table 2.1 of their patients (quasi-identifiers), except for their names (identifiers), then the attacker can link the data appearing in the last row of the table, implying identity disclosure: he knows the last row belongs to Susan. Also, because of this identity disclosure, attribute disclosure is followed: he learns that Susan suffered from a heart attack.

Person's name	**City**	**Age**	**Illness**
George	London	30	Cancer
Mary	London	30	Cancer
Paul	Manchester	60	AIDS
Anne	Manchester	60	AIDS
Susan	Manchester	58	Heart attack

Table 2.1: Table mimicking a database that enables identity and attribute disclosure

Case B. Identity disclosure without attribute disclosure

Now, let us consider that the attacker has the information: (*John, Manchester, 60, Heart attack)*. The variables released by the hospital are all the ones in Table 2.2 except for the

names. Then, the attacker may re-identify an individual (John), because the only possible record is the last one, but all the known attributes are used for re-identification and, thus, there is no attribute disclosure. Thus, this will be the case whenever all the attributes are needed for re-identification, or when the attributes not used do not cause disclosure.

Person's name	**City**	**Age**	**Illness**
George	London	30	Cancer
Mary	London	30	Cancer
Paul	Manchester	60	AIDS
Anne	Manchester	60	AIDS

Table 2.2: Table mimicking a database that enables identity disclosure without attribute disclosure

Case C. No identity disclosure but attribute disclosure

Now, let Table 1.3 reproduce the published data. Consider that the attacker has the information: (*George, London, 30*). Using this data, he will be able to learn that George has cancer, but identity disclosure has not taken place, because there are two people having the attributes (*London, 30*). Therefore, this is the case in which all indistinguishable records have the same value for a confidential attribute.

This is the case reached by a well-known privacy model for databases, *k*-anonymity**.**

Person's name	**City**	**Age**	**Illness**
George	London	30	Cancer
Mary	London	30	Cancer
Paul	Cambridge	60	AIDS
Anne	Cambridge	60	AIDS

Table 2.3: Table mimicking a database that enables attribute disclosure but not identity disclosure

A dataset satisfies ***k*-anonymity** with respect to a set of quasi-identifiers when every combination of these quasi-identifiers in the dataset appears at least for *k* different records [10].

2.1.2. Data privacy in unstructured data

The case of unstructured data, such as raw plain documents, is far more challenging and has received much less attention, despite being the most usual way of information exchange.

Their lack of structure avoids defining a priori sets of identifier and quasi-identifier attributes. Thus, to care for privacy protection in textual documents, two tasks should be performed:

i) Detection of textual terms that may lead to disclosure of sensitive information.

ii) Removal or obfuscation of those terms.

The second step may be carried out via two different methods:

a) Removing or blacking out the sensitive terms.

b) Obfuscating sensitive terms by replacing them with appropriate generalizations.

The latter approach is more advantageous, as it better maintains the utility at the output. Additionally, the presence of blacked-out parts can give a clue of the document's sensitivity to potential attackers.

The tool proposed in this project combines both blacking out and generalization techniques in a semiautomatic process. Also, it focuses in anonymity rather than confidentiality, that is, in identity disclosure protection rather than in attribute disclosure protection. Thus, the approach will search a goal similar to the one presented in the Case C for structured data: disclosure of a confidential attribute will not be compromising as long as it does not involve identity disclosure. The fact of having a raw text instead of a database will, as can be intuited, entail a considerably more complicated process regarding the definition of sensitive terms.

2.2. Information theory

"A Mathematical Theory of Communication" was published by the mathematician and engineer Claude Shannon in 1948 [11]. This paper implied an important transformation in people's understanding of information.

Before his publication, there was a rather poor conception of what information and the communication process were. In his paper, Shannon showed how information could be quantified with absolute precision, and stated that all information media had an essential unity: telephone signals, texts, radio waves, pictures and essentially every form of communication could be encoded in bits.

Thus, it became clear that information was a well-defined and, most importantly, measurable quantity. Certainly, as Shannon himself claimed:

A basic idea in information theory is that information can be treated very much like a physical quantity, such as mass or energy [12].

2.2.1. Information content

Consider a random process governed by a discrete random variable X which is completely defined by the set of values (possible outcomes) it can take from a finite set, χ , which we assume finite, and its probability distribution, $\{ p_X(x)\}_{x \in}$.

Suppose the probabilities of occurrence of each possible outcome, $\mathbb{P}(X = x_i) = p_X(x_i) = p_i$, are $p_1, p_2, \dots, p_n$. These probabilities are known quantities, but we know nothing else concerning which event will occur. Is there a measure to quantify how uncertain we are of the outcome, or, in other words, how much information the event produces?

It is intuitive that specifying that a high-probability event occurred does not transmit much information (the result was "expected"), while specifying the occurrence of a low-probability event transmits a considerable amount of information [13].

Therefore, the uncertainty or information regarding the occurrence of s_i should depend on its probability of occurrence, p_i, through a function $f(p_i)$ that increases as p_i decreases [14]:

In order to find a suitable expression for the function $f(p_i)$, some requirements should be imposed in order that the following properties of information are satisfied.

1. Information is positive or equal to 0:

$$I(x_i) \geq 0 \tag{1}$$

   Any event with $p_i < 1$ produces some amount of information, whereas an event occurring with $p_i = 1$ will produce no information at all.

2. Information is additive. That is, the information of an event being the intersection of two independent events must be the sum of the information resulting from the two independent events occurring.

Let us consider an event x_i composed of two different and independent events, x_{i1} and x_{i2} :

$$x_i = x_{i1} \cap x_{i2} \tag{2}$$

Regarding the additivity of information:

$$I(x_i) = I(x_{i1}) + I(x_{i2}) \tag{3}$$

And therefore:

$$f(p_i) = f(p_{i1}) + f(p_{i2}) \tag{4}$$

Because x_{i1} and x_{i2} are independent events, it is found that:

$$p(x_{i1} \cap x_{i2}) = p(x_{i1})p(x_{i2}) \tag{5}$$

So expression (4) may be reformulated as:

$$f\big(p_X(x_{i1})p_X(x_{i2})\big) = f\big(p_X(x_{i1})\big) + f(p_X(x_{i2})) \tag{6}$$

The class of function having this property is the logarithm function of any base. Because it decresases with probability and must be nonnegative whereas probabilities range from 0 to 1:

$$f(p_i) = -\lambda \log p_i \tag{7}$$

Where λ is a positive constant.

The unit of information was defined beginning from the most simple case: that of choosing one from two equally probable:

$$I(x_1) = I(x_2) = -\lambda \log \frac{1}{2} \tag{8}$$

By setting $\lambda = 1$ and using the logarithm in base 2, the *bit* could be defined:

$$I(x_1) = I(x_2) = -\log_2 \frac{1}{2} = 1\ bit \tag{9}$$

And, therefore, the expression that is today known as Shannon Information or **Information Content** appeared:

$$I(x_i) = -\log p_X(x_i) \tag{10}$$

Also found as:

$$I(x_i) = \log \frac{1}{p_X(x_i)} \tag{11}$$

The choice of the logaritmic base defines the unit for information measurment. If base 2 is chosen, then information will be measured in binary digits or *bits*. When using the natural logaritm of base e, the unit is the *nat*. If 10 is used, the units may be called decimal digits.

Shannon defined information from an engineering point of view, as himself stated [11]. He recognized that messages usually have meaning, that is, they refer to or are correlated in accordance with some system with certain conceptual entities. But,he said, these semantic aspects of communication were not relevant to the engineering problem: nature and meaning of data does not matterwhen it comes to information content. The important thing is that he actual message is *one selected from* the entire set of messages, so information must be quantified in terms of probability distribution and uncertainty.

2.3. Natural Language Processing

Natural Language Processing, commonly known as NLP, is the study centred in the interactions between human language and computers. It lays on the intersection of computer science, artificial intelligence and computational linguistics.

Apart from common text processors, which treat text simply as a sequence of symbols, NLP takes into consideration the hierarchical structure of language. Like this, several words make a phrase, several phrases make a sentence, and sentences at last convey ideas. NLP enables computers to analyse, understand and derive meaning from human language. It used for developers to perform tasks such as automatic summarization, relationship extraction, translation, named entity recognition, sentiment analysis and speech segmentation, among others [15].

There are two key approaches to Natural Language Processing: the statistical approach and the linguistic approach.

The **statistical approach**, characterized from each document's set of key words. Each term is assigned a weight depending on its importance, usually determined by the frequency of appearance within the document. Thus, the word order, meaning or structure is not taken into consideration.

The **linguistic approach**, in which different techniques explicitly encoding linguistic knowledge are applied. The morphological analysis is performed by taggers that assign a grammatical category to each word depending on the morphological characteristics found. For syntactic analysis, that is, the relation and grouping of words to form larger units such as phrases or sentences, parsers are applied. After the syntactical structure has been found, the aim is to extract the sentence meaning from it: the semantic analysis. For this, the most common tool to use is the lexicographic database WordNet which, as will be seen, is made up of synonym groups called *synsets* which provide short definitions as well as the different semantic relationships between synonym groups.

NLP algorithms are typically based on machine learning algorithms. That is, instead of hand-coding large groups of rules, it makes use of machine learning to automatically learn those rules by examining large amount of examples and making statistical inference. Normally, the model gains more accuracy when more data is analysed. Some examples of this will be seen throughout the project [16].

2.3.1. Natural Language Processing and Information Theory

It is common to use Information Theory concepts in NLP, and to make use of NLP in processes of information retrieval. That is, the information content provided by language may make use of its statistics, but meaning cannot be ignored in practice.

There are two main problems, however, when pretending to use NLP for obtaining information, which are inherent characteristics of NLP: linguistic variation and ambiguity [17].

Linguistic variation refers to the possibility of using different words and expressions to communicate the same idea. This may lead to the omission of terms due to having the same meaning but different statistics than others.

Ambiguity, on the other hand, is the fact that a word or phrase allows more than one meaning. This leads to noise, as terms with the same statistics may be treated equally despite having different meanings.

Whether there is or not any method that enables to avoid these two problems, which are inherent to language, may be further discussed.

2.4. Model performance concepts

Given a certain population, when each of its members belongs to one of a number of different classes, a classification rule may be required. That is, a procedure by which one is able to predict to which class or set each element belongs to. Thus, given a data set

consisting of a set of population elements and the classes they each belong to, a classification rule is a function that assigns each element to a predicted class.

In a **binary classification**, the label (i.e., the corresponding class for an element) can only take one of two values [18]. A multi-class classification, on the other hand, involves assigning an object to one of several classes. Some classifications rules consist of static functions, while others may make use of machine-learning algorithms.

A perfect classification is that for which every element in the population is allocated to the class it truly belongs to. When errors appear, the classification is imperfect and a statistical analysis must be applied.

2.4.1. Binary classification testing

Consider a binary classification, where a decision is required as to whether or not an element has a qualitative property or some specified characteristic. Thus, every item is predicted to belong to either the class possessing a certain property (Class = Yes) or to the class that does not (Class = No).

Let us take as an example a medical testing case, where one needs to determine if an individual has a certain disease or not. The classification property would be, in this case, the presence of the disease.

Given a classification of a particular dataset, four possibilities may be encountered regarding the combination of the actual data category and predicted category [19]:

		Actual class	
		Class = Yes	Class = No
Predicted class	Class = Yes	True positive	False positive
	Class = No	False negative	True negative

Table 2.4: The four possibilities regarding the combination of actual class and predicted class.

When the observations (class values) are correctly predicted, they are either True Positives or True Negatives.

- **True Positives (TP):** These are the correctly predicted positive values. Both the actual class and the predicted class have value “Yes”. It would be the case in which a patient is correctly diagnosed with the disease.
- **True Negatives (TN):** These are the correctly predicted negative values, which means both the actual class and predicted class have value “No”. In our example, it is the case of a patient not having the disease, and the test result confirming this fact.

When, on the other hand, the predicted class value contradicts the actual class value, prediction errors are encountered. These can be False Positives and False Negatives.

- **False Positives (FP)**: The wrongly predicted positive values, so prediction class value is “Yes” and actual class value is “No”. It is the case of detecting a disease when it is not present.
- **False Negatives (FN):** The wrongly predicted negative values. Thus, prediction class value is “No” and actual class value is “Yes”. It would be not detecting a disease when it is present.

2.4.2. Classification performance metrics

There are some useful and typically used metrics when evaluating the performance of a classification algorithm [19]:

- **Accuracy**: It is the ratio of correctly predicted observations to the total observations. It is thus computed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

- **Precision**: It is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

  It would answer the question “Of all patients labelled as sick, how many are actually sick?”. High precision relates to low false positive rate.

- **Recall (Sensitivity)**: It is the ratio of correctly predicted positive observations to all the observations in the “Yes” actual class.

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

  This metric would answer the question “Of all the patients that are actually sick, how many were labelled as sick?”.

- **F_β score**: It is used to measure the accuracy of the model using both precision and recall and depends on a parameter β, a real non-negative number related to how many times is recall more important than precision:

$$\beta = \frac{weight\ on\ recall}{weight\ on\ precision} \tag{15}$$

  Hence, β will take values from 0 to 1 if accuracy is decided to be more important, whereas values from 1 to ∞ are taken whenever recall must be more relevant.

$$F_\beta\ score = (1 + \beta^2)\frac{Precision * Recall}{(\beta^2 * Precision + Recall)} \tag{16}$$

The most used and intuitive is F_1 score, for being the harmonic mean between these two measures. In that way, both false positives and false negatives are taken into account. Other used measures are F_2 and $F_{0.5}$ scores.

$$F_1\ score = \frac{2 * (Recall * Precision)}{(Recall + Precision)} \tag{17}$$

Some performance measures may be preferable among the others depending on two key data set aspects [20].

1. The **class distribution**. That is, the ratio between positive values and negative values in the data. Most of the times, the class distribution is not 50/50.
2. The **cost** of wrongly predicted observations: false positives and false negatives may have different cost. As an example, the cost of administering a drug to someone who does not need it may be different than the one of not administering a drug to a patient that needs it.

Despite being the most intuitive performance measure, accuracy is only good for symmetric data sets, where the classes are almost evenly distributed and the cost of false positives and false negatives is roughly the same.

Both precision and recall are convenient measures if there is an uneven class distribution. Precision makes a quantification of how good predictions are with regard to false positives, whereas the recall measure shows how good predictions are with regard to false negatives. Whichever type of error is more relevant, or costs more, may determine the most adequate measure in each case.

F_1 score, on the other hand, works well for uneven distributions, but works best if false positives and false negatives have similar cost.

The recommended data set conditions to be met in order to use a measure or another are summarized in the table below.

		Class distribution	
		Even	Uneven
Cost	Higher for FN	Recall	Recall
	Equal	Accuracy or F1 score	F1 score
	Higher for FP	Precision	Precision

Table 2.5: Performance measure to use based on the dataset conditions

F_β score gives, of course, infinite intermediate options between the ones presented on the table by changing the parameter β according to the specific needs.

3. Tool development

In this section, we are going to comment how the tool has been developed. From the premise, to specifying the main goal, explaining the choice of the programming language and how the algorithm was developed, commenting the problems encountered and the solutions adopted.

3.1. Premise

The idea of this project initially took off as an upgrade of the already existing anonymization tool created by the SISCOM department of Universitat Politècnica de Catalunya (UPC) in the European project CIPSEC ("Enhancing critical infrastructure protection with innovative security framework"), the aim of which is to create a unified security framework [21].

The mentioned tool has the ability of sanitizing the data collected by companies in order to make it impersonal and, thus, available to be shared with third entities while complying with GDPR legislation. It works by receiving a JSON file as input and sanitizing it according to a specific policy. Because the policies are created by defining a series of regular expressions, the user is provided with the flexibility and freedom to sanitize data according to their needs. Thus, the output file contains the sanitized data as desired.

Once the state-of-the-art study was carried out, however, it was found that, despite being the usual way in which data is shared, textual data sanitization had received very little attention.

Structured data such as databases profit from the structure of data to easily find the sensitive attributes. That would be the case of, for example, the ID number, always composed of a definite number of digits and a letter. Anonymization of raw text, on the other hand, is far more ambitious due to the absence of structure in data and, thus, the difficulty in finding explicit identifiers.

All in all, the development of an anonymization tool for textual data seemed both more challenging and necessary. CIPSEC's anonymization tool relied on the structure of data and thus only detected as sensitive those fields previously defined as sensitive, ignoring other terms which are potentially sensitive due to their specificity.

3.2. Main goal

The tool's main goal is to anonymize raw text documents via sanitization methods: blacking out and generalizing. That is, to develop an algorithm able to detect and remove or generalize those terms that may disclose the identity of any individual that is referred to in the text.

As stated in section 2.1.2, this should be done by performing two different tasks:

1. Detecting those textual terms that may lead to disclosure of sensitive information. Because this project focuses on anonymity (identity disclosure prevention) rather than in confidentiality (attribute disclosure prevention), sensitive terms will be those disclosing an individual's identity.
2. Generalizing or blacking out those terms.

Now that the objective has been set, the question arises: How is one able to automatically detect those terms that are sensitive?

Sensitive terms may be defined as those that, because of their specificity, reveal more information than ordinary ones. If a potential attacker is assumed to have a certain level of information, sensitive terms will increase the information the attacker possesses.

For simplicity, terms will assume to be statistically independent from each other The information provided by each textual term, t_i, can be computed, recalling expression (10), from its occurrence probability $p(t_i)$ as:

$$I(t_i) = log_2 \frac{1}{p(t_i)} \tag{18}$$

Because of the independency assumption, this value is, in practice, un upper-bound value to the real one. In order to detect sensitive terms, we will rely on this expression and compute those terms that reveal an amount of information above the one desired.

This will be the basis of the algorithm, but contribution of NLP techniques will be also needed, as will be seen.

3.3. Programming language

To develop the tool, Python [22] has been chosen as the programming language. While it is true that this decision was initially motivated for the fact that the CIPSEC anonymization tool used this language, there were some key advantages that brought us to stick with it over the other languages.

Python is a free, high-level general-purpose programming language with a design philosophy that emphasises code readability. It supports multiple programming paradigms, including procedural, functional and object-oriented.

Python's syntax is simpler and its codes are shorter than other programming languages such as Java and C++, thus being easier to amend, rework and optimize. It is widely used, not only for small and medium level companies, but for leading ones such as Google, Spotify, Instagram and Dropbox, as well as non-IT companies such as NASA, Electronic Arts and Disney [23].

The main benefit in the case that concerns us, however, is the vast amount of modules and built-in libraries it contains, easing and speeding the code development. An example of this is NLTK (Natural Language Toolkit), a powerful Python platform including a wide variety of tools, libraries and algorithms for natural language processing, representing a

great use to this project. Thus, both Python and NLTK basic online courses have been taken to have a certain level of understanding. Specifically, Python 2.7.15 and NLTK 3.4.1 have been used.

3.4. Basic algorithm structure

Figure 3.1: Basic algorithm structure scheme

The tool is designed as follows:

1. A raw, unsanitized, individual-identifying text is received as input. This can be imported from a webpage or, more commonly, be in the computer in the form of a text file (".txt").
2. The input file is anonymized as a result of a process involving different techniques.
3. The output text is, thus, a file in text format containing the anonymized text to be saved by the user. Sensitive terms are either blacked out (substituted by "***") or generalized.

The anonymization process is the key to achieve the main objective and the process of developing it, as well as the different techniques involved need to be forthwith analysed and discussed.

3.5. Algorithm development

3.5.1. The corpus

In order to determine the occurrence probability of each term, an appropriate corpus is needed. That is, a large collection of texts that intends to represent the true distribution of terms of a natural language.

First, the **Brown Corpus** was considered. It was created in 1961 at Brown University, becoming the first million-word electronic corpus of English. It contains texts from 500 different sources, each categorized by genre such as news, editorial, reviews, religion, fiction and many more [24]. NLTK provides NLTK corpus readers, a package providing modules with functions that enable reading corpus files in a variety of formats. The Brown Corpus is one of the many corpora distributed in this NLTK package. By using one of the various functions, one is able to obtain a list of all the words in the corpus. Moreover, with the Frequency Distribution function, the frequency of each vocabulary item in the corpus is found; that is, the number of occurrences of a particular term in

the whole corpus. Thus, by knowing the total number of terms in the whole corpus, the occurrence probability and, therefore, the information content of each term may be found from Equation 18.

Despite appearing as a good choice in the first place, an import drawback has been found when using the Brown corpus in the algorithm: due to the fact of being created many decades ago, relatively modern terms have a considerably low frequency. The term “video”, for example, appears a number of 2 times out of a total of 1161192, which is not quite accurate regarding nowadays’ used English.

When searching for an alternative, **Datamuse API** was found. Datamuse API is The Datamuse API is a word-finding query engine for developers [25]. The endpoint “/words” is used to find list of words that match a given set of constraints, such as on meaning, spelling or sound. Thus, query parameters include “ml” (means like), “sl” (sounds like) and “sp” (spelled like), among others. In our case, the constraint “sp” is the desired one. Moreover, metadata (“md”) flags, consisting of a list of single-letter codes, are used for requesting additional lexical information regarding the query. The one we are interested on is “f”, for Word frequency, which gives as value the number of times the word or multi-word phrase occurs per million words of English text according to Google Books Ngrams. Google Books Ngrams is an online search engine that charts the frequency of any set of strings in sources printed between 1500 and 2008 in Google’s text corpora in various languages [26].

Thus, when the input text is read, the first thing to do is to *tokenize* the text using the NLTK word tokenizer. That is, to divide larger strings into lists of substrings that are, in this case, words. Subsequently, a function has been created that enters each word as a query in Datamuse API, by asking to read the obtained results when searching the following url: http://api.datamuse.com/words?sp=xxxxx&md=f&max=1 , where “xxxxx” is replaced each time by the word in question. The resulting information is obtained in JSON format, and looks like:

```
[{"word":"cat","score":6707,"tags":["f:38.386177"]}]
```

The value we are interested in is the one preceded by “f:”. In order to extract this value, the Regular Expression module for Python is needed. In this case, we ask for the string pieces matching a pattern consisting of the one composed by an “f” followed by a colon and an indefinite number of digits with an optional period in between. In this way, the frequency value is obtained for each word entered as input. The probability of occurrence for each term may be obtained by simply dividing this value by 10^6.

An important disadvantage has been faced regarding the use of Datamuse: the great amount of time it takes for the tool to make the query and get the results for each of the different words in the text, which jeopardizes the tool overall speed. To solve this, the idea of creating local dictionaries came upon. That is, to write in a text file all the words and their associated frequencies for every text entered as input. Thus, when one wants to anonymize a new text and the terms’ frequencies need to be looked for, the tool first checks if the term in question is already present in the local dictionary, which

is a local text file. If it is, it takes as the frequency value the one established in the local text file. If it cannot be found, the word is recognized as a new one, and a query is made to Datamuse API to find the frequency value. When the process finishes, all the new word-frequency tuples are written in the local dictionary (see Figure 3.2). Thus, the more texts are anonymized, the more words are already present in the local dictionary. Because reading a text file offline is much faster than making a query and waiting and reading for the result, the execution time is greatly reduced.

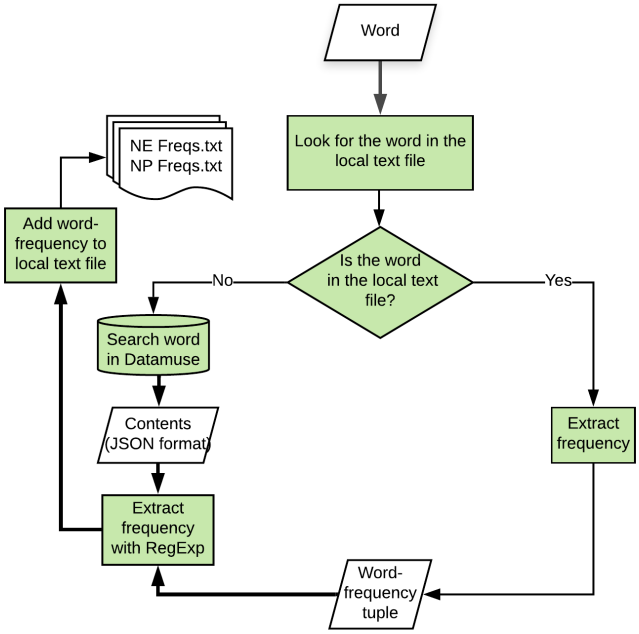

Figure 3.2: Flowchart illustrating the use of local dictionaries and Datamuse database to obtain word frequencies.

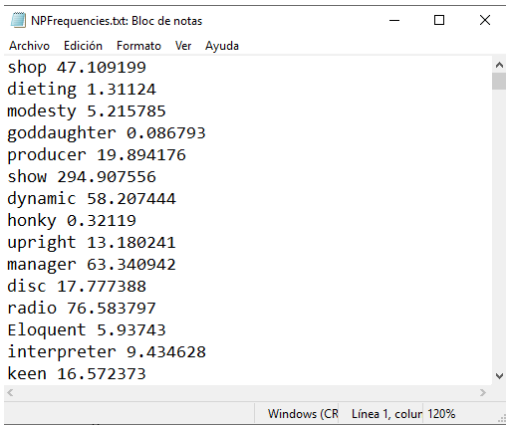

Figure 3.3: Example of the local file text containing the word-frequency tuples.

Datamuse's service can be used without restriction for up to 100.000 requests per day. Although that may be a limitation in extreme cases, where great volumes of texts have to be anonymized, the creation and continuous actualization of the local frequency dictionaries make it unlikely to reach that limit.

3.5.2. Part of Speech tagging

The assumption made so far is that a term's sensitivity is only dependent on its probability of occurrence. This was what Shannon proposed when defining information; that the meaning is irrelevant and therefore does not need to be considered when computing the amount of information provided. This is also what proposes the statistical approach for NLP.

Using a text from Wikipedia a qualitative test was done, the frequency value chosen randomly, in order to check how the tool performed overall. Let us check the anonymization of a sentence extracted from the text, where the sensitive terms, namely, those with a frequency below the one specified in the code as threshold, have been blacked out:

Original sentence: *Returning to Los Angeles , he enrolled at Los Angeles City College in 1965 .*

Sentence after anonymization: **** to Los *** , he *** at Los *** City College in 1965 .*

It can be seen that the verbs, "returning" and "enrolled", have been blacked out, while nouns such as "City" and "College" remain in clear form. While it is true that the first words may be more "unusual", one can easily note that *where* this individual enrolled reveals much more information of him than the fact that he *enrolled* somewhere.

Therefore, the assumption that information content must only depend on statistics happens not to give the best results when put into practice, especially in the field of linguistics where morphological, syntactic and semantic characteristics have an important role, and should therefore be taken into account aside from probability.

Inspection of other examples led to the decision to take a different approach: only noun phrases (optional adjective + noun) and proper nouns should be considered as potential sensitive terms. Verbs, adverbs, prepositions and other grammatical categories are not considered to be likely of revealing important information of an individual.

In order to do that, a **part-of-speech tagger**, or POS-tagger, must be used. Part-of-speech tagging is the process of classifying words into their parts of speech (commonly known as word classes or lexical categories) and labelling them accordingly. NLTK's default recommended POS tagger is a machine learning algorithm based on the *tagset* (tag dataset) Penn Treebank [27]. The obtained result when using POS-tagger is a word-tag tuple. Let us see an example with a very simple sentence:

```
[('I', 'PRP'), ('like', 'VBP'), ('brown', 'JJ'), ('cats', 'NNS'), ('and', 'CC'), ('black', 'JJ'), ('dogs', 'NNS')]
```

A list of tuples is obtained, each word paired with its corresponding tag associated with a grammatical category. Like this, "PRP" is used for tagging a personal pronoun, "VBP" for verb, non-3rd person singular present, "JJ" for adjective, "NNS" for a plural noun and "CC" for coordinating conjunction.

Because we are interested on detecting both proper nouns (singular and plural) and noun phrases (optional adjective plus singular or plural common noun), NLTK's regular expression parser must be used. A parser analyses strings of symbols conforming the rules of formal grammar. *RegexpParser* uses a set of regular expression patterns to specify the behaviour of the parser; that is, grammar is defined with regular expressions that determine POS tags. It gives a tree as a result. Let us look at the previous example when *RegexpParser* is applied in our case.

```
(S
 I/PRP
 like/VBP
 (NP brown/JJ cats/NNS)
 and/CC
 (NP black/JJ dogs/NNS))
```

It can be seen that the parser distinguished the Noun Phrases, NP. From this, lists collecting all of both proper nouns and noun phrases in the texts are created. As before, the frequency of these words was analysed and, if lower than the threshold frequency, they were blacked out from the text. Let us analyse the result with an extracted sentence:

Original sentence: *His father, Charles Robert Hardy Douglas Andrews, born in Effingham, Kansas, was a newspaperman, pioneering radio soap opera writer, novelist, and screenwriter.*

Sentence after anonymization: *His father, Charles Robert *** *** ***, born in ***, ***, was a newspaperman, *** radio *** *** writer, ***, and ***.*

This result is clearly improved with respect to the one obtained earlier. However, the fact that a state, "Kansas" is anonymized before part of a person's name is considerably risky. That is, while proper nouns referring to locations, for example, may be quasi-identifiers, a person's name may be able to unequivocally disclose identity.

Therefore, the aim was to find a technique which is able to distinguish between different entities to the which the text refers.

3.5.3. Named Entity Recognition

Named entities (NE) are definite noun phrases that refer to specific types of individuals, locations, facilities and dates, among many others. In order to detect them, a **Named Entity Recognition** system is used.

NLTK has a default named entity chunker (NEC), which identifies all the chunks (substrings of texts that cannot overlap) referring to named entities [28]. It detects the following named entities:

- Persons
- Organizations

- Facilities
- GPE (Geo-political entities)

NLTK's NEC works by making use of a supervised machine algorithm known as MaxEnt classifier, which gets its name from *maximum entropy.* That is, for a given set of conditions, the algorithm searches for the maximum entropy, which is obtained when the discrete probability distribution is uniform [29]. The model has been trained on data from a corpus that has been manually annotated for NEs.

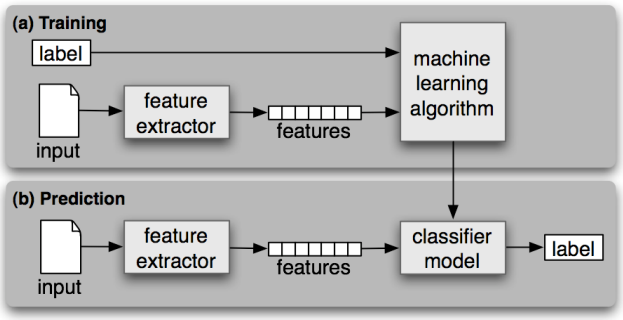

Figure 3.4: Training (a) and prediction (b) in machine learning algorithms

In order to optimize the performance of a machine learning algorithm, the features that work best should be chosen. The features are those characteristics on the which statistics are computed in order to predict the classification. For NLTK's NEC, features include the shape of the word, the length of the word, the word itself, the POS tag for the word, the POS tag of both preceding and following words, or even the shape of the word combined with the tag of the following word [30].

Before using the NEC, words have to be tokenized and POS-tagged, as in the previous case. Let us see a demonstration:

```
(S
  (PERSON Steve/NNP)
  (PERSON Jobs/NNP)
  was/VBD
  born/VBN
  in/IN
  (GPE California/NNP)
  ./.)
```

Entities referring to both persons and GPE, in this case a state, are tagged correctly.

Now, the approach the tool proposes is the following:

- Named Entities referring to PERSONS will be directly blacked-out from the text. That is, because there is a certain risk that they may be unequivocal identifiers, they must be removed independently from their statistics.

- The other type of named entities (GPE, organizations and facilities), as well as noun phrases, will be or not blacked-out regarding their occurrence probability, given by their frequency of apparition.

Let us study another example, with a more complex sentence than the previous one.

```
(S
  when/WRB
  (PERSON Jobs/NNP)
  was/VBD
  just/RB
  21/CD
  ,/,
  he/PRP
  and/CC
  (PERSON Steve/NNP Wozniak/NNP)
  started/VBD
  (PERSON Apple/NNP Computer/NNP)
  in/IN
  (ORGANIZATION Jobs/NNP)
  's/APOS
  family/NN
  garage/NN
  ./.)
```

This shows that NEC, despite detecting all the named entities, may classify them wrongly. “Jobs”, which is a person, is classified both as a person and as an organization, and will therefore be in both “persons” and “organizations” lists. “Apple Computer”, an organization, is classified wrongly as a person in this sentence, and correctly as an organization in other sentences in the text, so will be therefore in both lists too. In this privacy protection framework, it is riskier that a person’s name is not directly anonymized, than an organization being directly anonymized when it should not be.

It is for this reason that, whenever an entity is classified as two different entity types, one of them being “persons”, it will be excluded from the other entity type list and thus only belong to “persons”, therefore being directly removed. In this way, we assure that all persons’ names are blacked out, with the possibility of blacking out entities that are not sufficiently sensitive as a drawback.

Let us analyse the output when this new approach is employed. An arbitrary threshold frequency of 50 (occurrences per million) has been used, meaning terms are considered as sensitive when their occurrence probability is below $5x10^{-5}$.

Original fragment: *Andrews ' mother was Irene Colman , an actress of French-Canadian descent born in Nashua , New Hampshire . She played a chorus girl in several Gold Diggers movies and had ingenue roles in a number of other movies .*

Fragment after anonymization: **** ' mother was *** *** , an *** of French-Canadian *** born in *** , New *** . She played a *** girl in several Gold *** *** and had ingenue roles in a number of other *** .*

It can be seen that the result improves regarding privacy. All the entities referring to persons are blacked out, plus many quasi-identifiers such as *Nashua* (city) are blacked out. Let us focus on the term *chorus*, followed by *girl,* the first one being blacked out. The fact that this person appears in several movies as a *girl* gives a certain amount of information, while specifying that the role she played was that of a *chorus* girl discloses a greater amount of information and may even be a potential quasi-identifier.

It should be noted, however, that parts of named entities, such as *New* or *Gold* in this example, have not been blacked out. In this case, the fact that the mother of the individual appeared in some entity starting by *Gold* should reveal more information than the fact that she appeared in *movies*, which is intuitive regarding the context and has been nevertheless blacked out.

Inspection of anonymized texts led to the observation that this fact is concurrent and motivates the apparition of a new hypothesis: that named entities, just because of their nature, are likely to disclose more information than noun phrases. This suspicion is also encouraged by the fact that Datamuse API is case-insensitive, that is, it does not distinguish between uppercase and lowercase letters of the words queried, thus treating equally in terms of statistics the words *gold* and *Gold*, being the latter part of a named entity. Because their statistics are the same, they will be (or not) removed evenly. This new assumption will be further studied when making the evaluation of the tool in section 4.

3.5.4. Solving ambiguity: User interaction

To this point, it has been shown that applying information theory concepts is far from straightforward in the field of linguistics, where meaning plays an important role in practice and therefore one cannot rely only on statistics.

Moreover, the text context, interpretation and nuances of words also make influence, and are yet characteristics that go beyond any algorithm's capacity. Today, there is doubtfully any machine with the capability of identifying irony, polysemy or other tasks as efficiently as humans.

It is due to this fact that the idea of an upgrade that considers the interaction of the user for taking certain decisions came up. In this point, the tool turned from being automatic to semi-automatic, slowing down the anonymization process in exchange for a more accurate performance.

The first objective regarding this update is to solve ambiguity (seen previously in section 2.3.1) of noun phrases. Named entities, for being specific types of noun phrases or real world objects with proper nouns, are very unlikely to produce ambiguity. The approach works as before when selecting the noun phrases that are considered sensitive: after being detected as NP, their frequency is checked to see if it is lower than the one

specified as threshold. The difference is that, instead of directly blacking out these terms in the text, they are considered as *potentially* sensitive, and it corresponds to the user to decide whether they actually are sensitive, and thus should be removed, or if they can be left out in clear form.

Thus the user is given, for each potentially sensitive word, both the word itself and the sentence it appears in, to provide the context in question. Then, the user is asked to decide whether the word is actually sensitive and therefore has to be blacked out from the text or, on the contrary, it can be left in clear form. The user may answer these requirements with the keyboard, "y" for Yes, "n" for No, entering the answer directly in the Python Shell.

3.5.5. Generalization and WordNet

In section 2.1.2, two different ways of removing sensitive terms were seen: blacking out and generalizing.

So far, the sensitive noun phrases have been blacked out, thus reducing their amount of information to zero. What if we could generalize the terms labelled as sensitive, so that their amount of information turns to be below the threshold, but still conserving some information, and therefore, utility?

In order to fulfil that, the lexicographic database WordNet, included in the NLTK library, has been used. WordNet is made up of synonym groups called *synsets,* which are disposed in a hierarchal way, as will later be seen.

Let us navigate in WordNet with the help of an example to see and understand the possibilities it offers. Consider the word "motorcar". When we ask which *synset* it belongs to, we obtain:

```
[Synset('car.n.01')]
```

In the case of "motorcar" there is only one meaning, corresponding to the first noun sense of car. "`car.n.01`" is a *synset*, which contains a set of synonymous word or "lemmas". Let us look at the set of lemmas in this *synset*:

```
[u'car', u'auto', u'automobile', u'machine', u'motorcar']
```

It can be seen that all the words composing the *synset* are synonyms. Each word in the *synset* can have several meanings, but we are only interested on the single meaning that is common to all the words of the *synset* `car.n.01`. Let us see what is the definition proposed for this *synset*:

```
a motor vehicle with four wheels; usually propelled by an internal
combustion engine
```

Another feature available in WordNet is finding the hypernyms and hyponyms of words. This is possible because of the concept hierarchy in WordNet.

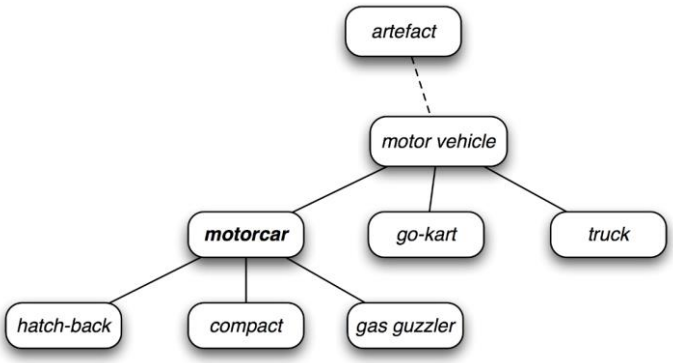

Figure 3.5: Example of the concept hierarchy in WordNet

A small fragment of a hierarchy example is shown in figure 3.4, where nodes are representing *synsets* and edges represent hypernym/hyponym relations with other *sysnsets*, that is, the relation between superordinate and subordinate concepts.

Let us analyse this hierarchy with the example of "motorcar", contained in the *synset* `car.n.01` . First, the hyponyms relations:

```
[Synset('ambulance.n.01'), Synset('beach_wagon.n.01'),
Synset('bus.n.04'), Synset('cab.n.03'), Synset('compact.n.03'),
Synset('convertible.n.01'), Synset('coupe.n.01'),
Synset('cruiser.n.01'), Synset('electric.n.01'),
Synset('gas_guzzler.n.01'), Synset('hardtop.n.01'),
Synset('hatchback.n.01'), Synset('horseless_carriage.n.01'),
Synset('hot_rod.n.01'), Synset('jeep.n.01'), Synset('limousine.n.01'),
Synset('loaner.n.02'), Synset('minicar.n.01'), Synset('minivan.n.01'),
Synset('model_t.n.01'), Synset('pace_car.n.01'), Synset('racer.n.02'),
Synset('roadster.n.01'), Synset('sedan.n.01'),
Synset('sport_utility.n.01'), Synset('sports_car.n.01'),
Synset('stanley_steamer.n.01'), Synset('stock_car.n.01'),
Synset('subcompact.n.01'), Synset('touring_car.n.01'), Synset('used-
car.n.01')]
```

And the hypernyms:

```
[Synset('motor_vehicle.n.01')]
```

Of course, features as the lemmas contained in each *synset* of the hyponyms or hypernyms could be now extracted, as well as the definitions.

All these concepts and elements in WordNet are indispensable for this new upgrade, for which the interaction of the user is again required. Once more, it has to be taken into account that named entities do no admit generalization and, thus, are blacked out without question when considered potentially sensitive.

Once the potentially sensitive noun phrases have been detected the user has to select, first of all, if they really are sensitive in the context they appear, as commented in the previous section. If the user decides they are, two options are given: blacking out the term or looking for its generalization. If the second option is chosen, then the word will be searched in WordNet and its several meanings (that is, the several *synsets* the word

may belong to) will be printed out along with their definitions. Then, it is the user's duty to select the meaning that suits best in the context. There is the possibility, however, that the potentially sensitive word is not in WordNet, which would imply blacking it out.

Once the corresponding *synset* for the word has been chosen, their hypernyms are searched. The user is offered the several options available concerning the hypernym s*ynsets* and, when the choice is submitted, the set of lemmas contained in the corresponding *synset* are also offered to the user to choose from so that the generalization conserves as much semantics and, thus, utility, as possible.

3.5.6. Graphical User Interface (GUI)

It has been seen that the tool has turned from being fully automatic to requiring the user interaction in order to take certain decisions that help to conserve utility. Because interacting by pressing letters from the keyboard and entering them directly in the Python Shell is a rather poor manner of interacting, as well as completely non-intuitive and even difficult for someone who is not familiar with the programming language environment.

Therefore, in order to provide a user-friendly, easy to interact with and more aesthetic tool, a Graphical User Interface has been developed with the help of Tkinter, a Python library created for this purpose. It will be showed and carefully explained along with the other functionalities of the anonymization tool in section 5.

All the Python code files used throughout the project realisation are appended at the end of the memory, in Appendix III: Python code files.

4. Tool evaluation

In this section, the evaluation of the tool takes place. The dataset is presented, as well as the procedure and results obtained for two different approaches used.

4.1. Evaluation dataset

In order to make the evaluation of the tool, five different raw texts have been picked. Because the aim is preventing a potential attacker from reidentifying an individual appearing in the text, Wikipedia biographies written in English were thought of as a good option.

With the purpose of making the setting as realistic as possible, some conditions have been searched for when selecting the persons' biographies.

Firstly, the person whom the text refers to is more desirably not to be well-known. That is, if the individual is very famous, the potential attacker may already have all the information contained in the text, and the anonymization process would thus be useless and make no sense. The tool is intended for ordinary people. Moreover, it is a non-realistic case as terms referring to certain attributes or anecdotes that would be, in principle, non-sensitive, could unequivocally identify a person whom these characteristics are known to be directly related to. As an example, the teeth shape, sexual orientation or suffering from HIV combined with the fact of being a singer may unequivocally identify Freddie Mercury.

Additionally, heterogeneity with regards to the people's work field was also taken into account for the sake of generality.

In the following table some of the text properties can be found. All the text used can be found in Appendices I.I.

	Name	**Work field**	**Text length (words)**
Text 1	Robert Adley	Politician	422
Text 2	Dan Amrich	Journalist Video games critic	1249
Text 3	Colman Andrews	Culinary critic	1024
Text 4	Eloise Gerry	Scientist	422
Text 5	Myrtle Allen	Chef	647

Table 4.1: Properties of the texts used in the tool evaluation

4.2. Procedure

It has to be noted that the evaluation is carried out for the totally-automatic version of the tool. The process involves, on one hand, anonymizing the biographical texts

automatically with the tool. This step has been carried out following two different approaches that will later be discussed.

Subsequently, an evaluation of the performance has been executed by computing and analysing the performance metrics seen in section 2.4.2. In order to make this evaluation, the texts have been previously anonymized by hand by conscientiously blacking out all the terms (and not only Named Entities or Noun Phrases) that are thought of as potential identifiers or quasi-identifiers. All the by-hand anonymized texts can be found in Appendices I.II.

The classification property is, in our case, **term sensitivity**. The blacked-out terms in the by-hand anonymized text are considered to belong to the "Yes" class value, for possessing the condition of sensitivity. The terms that have remained in clear form, on the other hand, are considered as non-sensitive, and are hence part of the class that does not have this property, Class = No.

Thus, a word-by-word comparison is made so as to check the combination of the predicted label and actual label of each term. Those terms that are correctly blacked-out by the tool are counted as True Positives, whereas the ones that are left untouched in both texts are counted as True Negatives.

When the tool blacks out a term that is in clear form in the by-hand anonymized text, however, a False Positive is encountered, and, contrarily, if it leaves a term untouched and it is supposed to be blacked-out, a False Negative is detected.

4.2.1. Approach 1

In the first approach, both Named Entities (NE), excluding persons, and Noun Phrases (NP) with a frequency (in occurrences per million words) below the threshold frequency have been anonymized. Named Entities classified as persons have been directly anonymized, regardless of their frequency, due to being clear identifiers. For the threshold word frequency, values ranging from 0 to 50 in steps of 1 have been taken. Thus, 50 anonymized texts (one for each frequency threshold) have been obtained for each biographical text.

Subsequently, evaluation of each one of the output texts with respect to the by-hand anonymized one, which contains the actual labelled data, has been carried out.

4.2.2. Approach 2

This second method relies on the basis that a Named Entity, because of its nature, is *per se* more sensitive than a noun phrases, as was introduced in section 3.5.3. Because Datamuse API is case-insensitive, the same frequency will be obtained for a word regardless of it being classified of a Named Entity or not.

Consider, for example, the terms "apple" (common noun referring to the fruit, tagged as NP), and "Apple" (multinational company, tagged as NE). When making a query in Datamuse for both word frequencies, the same value is found to be obtained: 19.314666

occurrences per million words. Thus, whenever we have a threshold frequency above this one, both terms will be anonymized (losing utility, as the term “apple” is usually unlikely to be sensitive) and, when the threshold frequency has a value below this one, both terms will be left in clear form (jeopardizing the privacy if the term “Apple” is considered sensitive in the context), having no intermediate option.

It is due to the inexistence of this possibility that the idea of a new approach came up. It intends to solve the problem by setting different threshold frequencies for Named Entities than for noun phrases, being the first ones of greater value than the second ones for, as has been stated, being more likely to disclose identifying information.

Hence, values for NE threshold frequencies have been picked ranging from 1 to 200 in steps of 1, and for NP values from 0 to 20 have been taken, also in steps of 1. The terms tagged as persons, as always, have been directly blacked-out.

In this way, the combination of each the values for both frequencies has led to the creation of 4000 automatically anonymized versions for each biographical text, each one being analysed for the counting of false/true positives/negatives and the performance measures obtained from them.

A summary of both approaches can be found in the table below.

		Named Entities (NE) anonymization		**Noun Phrases (NP) anonymization**
		Persons	**Others**	
Approach 1	Frequency range	All	0 - 50	0 - 50
	Step		1	1
Approach 2	Frequency range	All	0 - 200	0 - 20
	Step		1	1

Table 4.2: Parameters used in approaches 1 and 2

4.3. Results and discussion

Python’s libraries Matplotlib, Numpy and Pyplot were used to plot the results.

In order to make a proper discussion of the obtained results, the dataset characteristics should be analysed, as stated in section 2.4.2.

	Actual positives		**Actual negatives**	
	Total	**Percentage**	**Total**	**Percentage**
Text 1	63	14,89 %	359	85,11 %
Text 2	213	17,05 %	1036	82,95 %
Text 3	163	15,92 %	861	84,08 %
Text 4	56	13,27 %	366	86,73 %
Text 5	110	17 %	537	83 %

Table 4.3: Actual positives and actual negatives of the by-hand anonymized texts

Because, as can be seen from Table 4.3, all of the anonymized biographies have unevenly distributed classes, being the amount of negative class elements much larger, accuracy turns to be a non-reliable performance measure.

As can be seen from its expression, accuracy gives the total number of correct predictions out of the total elements. Why is it an inadequate performance measure in the case of uneven datasets? Let us take as example the case of Text 1. If our tool failed to detect any of the sensitive terms and, thus, labelled all the words in the text as non-sensitive, we would have null true positives and as much true negatives as actual negatives there were. Thus, for a tool that could not detect a single sensitive word, an accuracy value of 85,11 % would be obtained, being totally misleading. It is for this reason that accuracy has not been taken into account into the set of performance measurement metrics.

Regarding the wrong predictions cost, the privacy-utility compromise should be taken into account in every case, as each framework has its own privacy and utility requirements. It is for this reason that, apart from precision and recall values, which are related to false positives and false negatives, respectively, F1 score will also be computed for being the harmonic mean of these two.

Because privacy protection is the project's main concern, false negatives could be considered as more "risky" than false negatives. That is, leaving in clear form a word that is supposed to be anonymized is less desirable than anonymizing a word that is not supposed to be. Thus, F_β score with $\beta = 2$ will be an adequate measure to check the model's performance considering that recall is more important than precision and its weight should be bigger (2 times greater).

Due to the fact that five different biographical texts have been used in the evaluation and, for each of them, several performance measures are analysed, a considerable amount of plot figures have been generated.

Thus, in order to ease the reading of the project, one text will be treated as representative example, and the figures corresponding to all the other texts will be appended at the end of the memory, in Appendices II. A table summarizing the performance measures computed for each text will be created to enable a more general discussion.

4.3.1. Approach 1

The results obtained from the performance evaluation using the first approach have been represented in two-dimensional plots.

As it just has been stated, the exhaustive analysis of results will be carried out for just one text, which has been arbitrarily chosen to be Text 2 (data obtained for Text 1 slightly differs from the rest, as will be seen).

The false positives and false negatives percentages (with respect to the total text length) have been plotted together with the total errors in Figure 4.1. As could be intuited, the number of false positives rises when the threshold frequency increases, whereas the

false negatives tend to decrease. The total error plot has also the tendency to increase when the threshold frequency does, following a trend similar to the one of false positives.

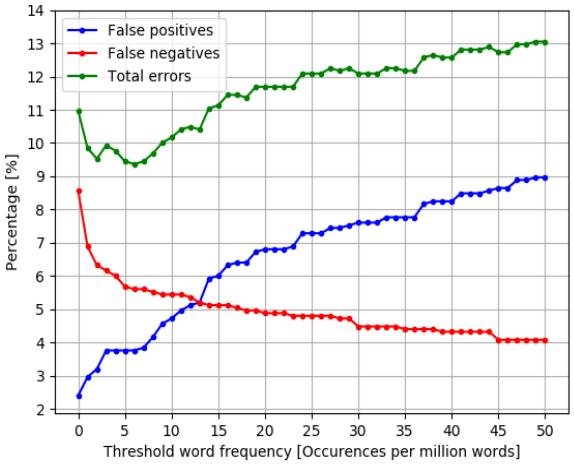

Figure 4.1: Plot of false positives, false negatives and total errors obtained for Text 2 using Approach 1

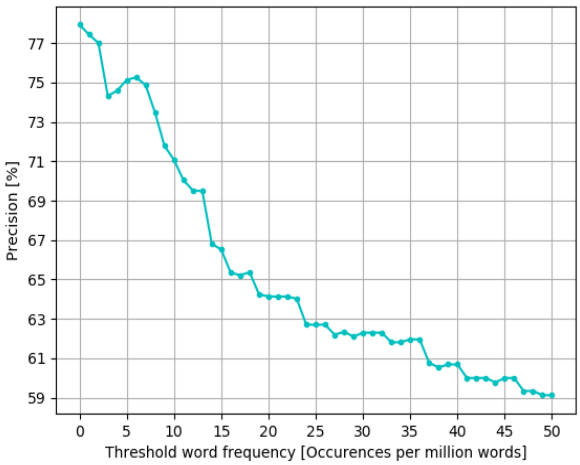

Figure 4.2: Plot of the precision obtained for Text 2 using Approach 1.

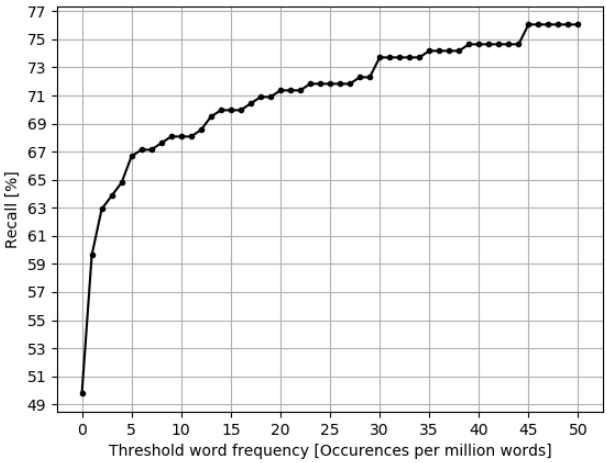

Figure 4.3: Plot of the recall obtained for Text 2 using Approach 1

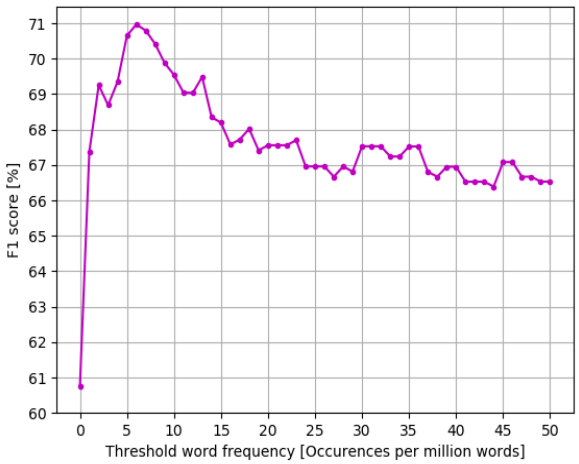

Figure 4.4: Plot of the F1 score obtained for Text 2 using Approach 1

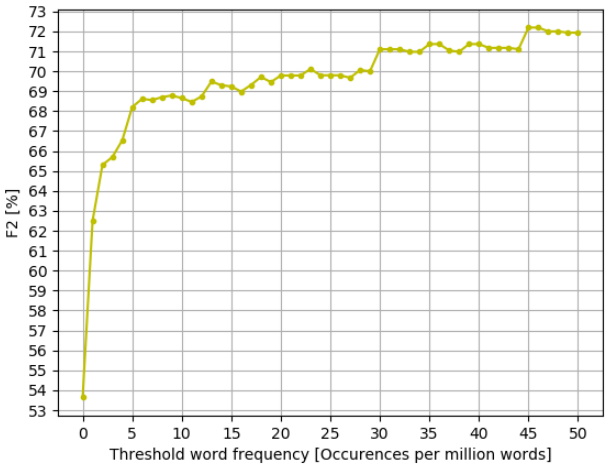

Figure 4.5: Plot of the F2 score obtained for Text 2 using Approach 1.

Because it is inversely related to the false positive rate (see Equation (13) precision, plotted in Figure 4.2, is usually a decreasing function. That is, for each frequency step increased, usually more false positives are acquired than false negatives are lost, and so the ratio of correctly labelled sensitive terms to all terms labelled as sensitive tends to decrease. Note, however, that in particular frequency step augmentations, the true positives gained amount is bigger than the false positives one and, thus, an increasing function plot is encountered.

Recall (Figure 4.3), on the other hand, gives information about the number of correctly predicted sensitive elements out of the total terms that are actually sensitive (see Equation 14). Thus, the higher the frequencies are, the higher the recall value will be. Because the actual positives in the denominator have constant value and true positives can only increase with frequency, recall, in contrast to precision, can only be an increasing function.

The fact that F1 score, plotted in Figure 4.4, tends to decrease when the frequency threshold increases enforces the idea that false positive rates prevail over false negative ones.

F2 score, plotted in Figure 4.5, on the other hand, increases with frequency due to giving a bigger weight to recall.

	Precision maximum		**Recall maximum**		**F1 score maximum**		**F2 score maximum**	
	Thres. Freq.	Value	Thres. frequency	Value	Thres. Freq.	Value	Thres. Freq.	Value
Text 1	0	97,22 %	49-50	63,51 %	1	70,08 %	16-18	64,42%
Text 2	0	77,94 %	45-50	76,05 %	6	70,96 %	45-46	72,19%
Text 3	0	92,85 %	36-50	64,42 %	2	65,11 %	36	63,83%
Text 4	3-4	76,92%	47-50	39,28%	3-4	48,78%	47-50	40,29%
Text 5	0	91,53 %	47-50	61,82%	1	67,43 %	47	61,82%
Mean	0,7	87,29%	47,4	61,02%	2,7	64,47%	38,8	60,51%

Table 4.4: Threshold frequencies and values of measures maxima for the five texts using Approach 1

Values in Table 4.4 give evidence the fact that F1 tends to be a decreasing function, its maximum being in every case at a very low frequency threshold, which confirms that the tool has an overall better performance at low frequency thresholds considering recall and precision as equally important measures.

4.3.2. Approach 2

The hypothesis that a Named Entity discloses more information than a Noun Phrase does for the same frequency (occurrences per million words) is the basis of this approach. To avoid redundancy due to the fact that false positives and false negatives rates have shown to increase and decrease in a certain manner, they will not be plotted.

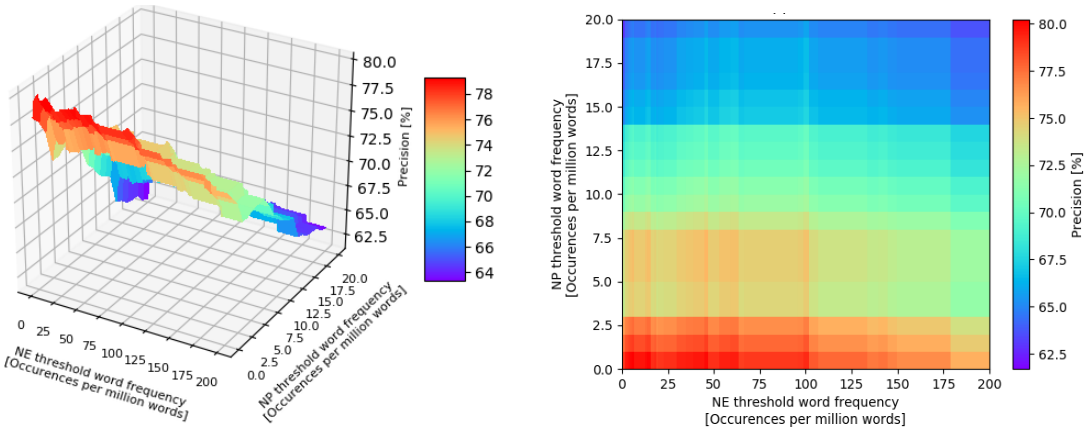

Figure 4.6: 3D (left) and 2D (right) surface plots of precision obtained for Text 2 using Approach 2

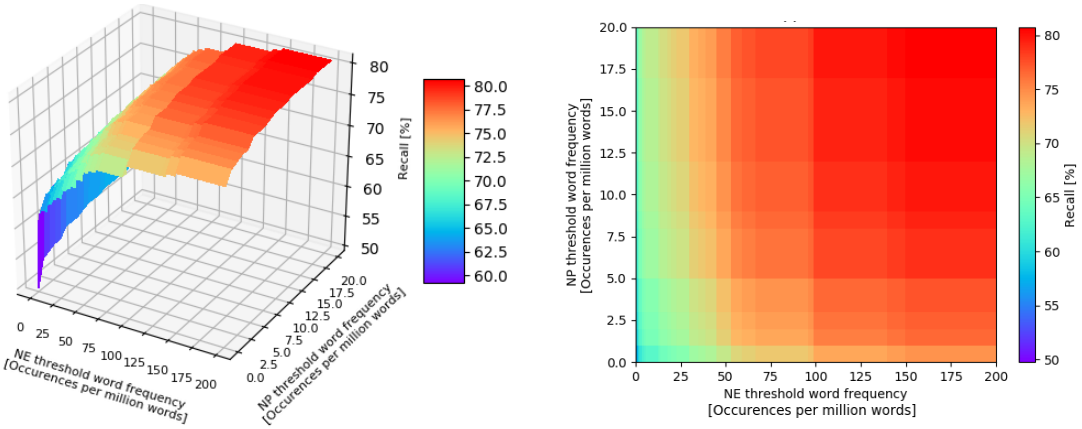

Figure 4.7: 3D (left) and 2D (right) surface plots of recall obtained for Text 2 using Approach 2

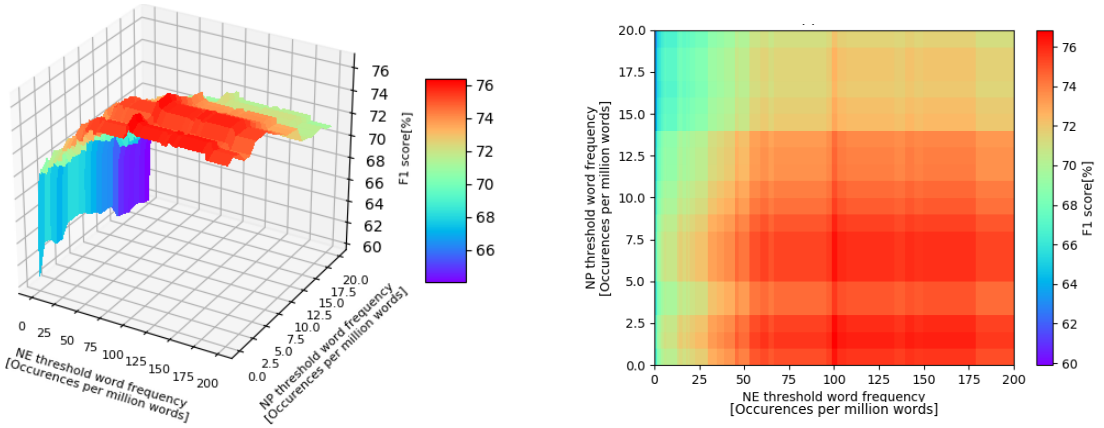

Figure 4.8: 3D (left) and 2D (right) surface plots of F1 score obtained for Text 2 using Approach 2

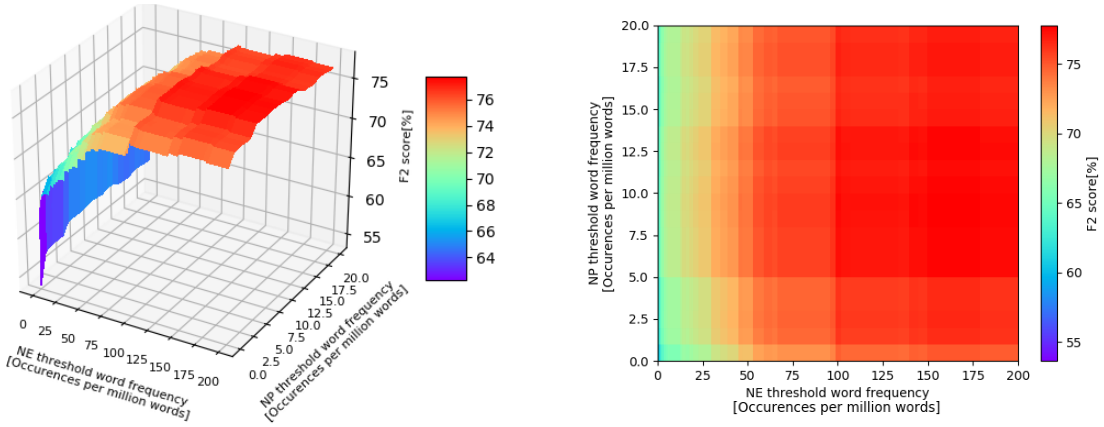

Figure 4.9: 3D (left) and 2D (right) surface plots of F2 score obtained for Text 2 using Approach 2

As can be deduced from Figure 4.6, precision gets higher values for NP frequencies close to 0 and for low NE frequencies, where the rate of false positives tends to be low. The opposite takes place for recall values, in Figure 4.7, that reach their maximum value when both frequencies are high.

F1 score, on the other hand, plotted in Figure 4.8, has greater values for mid-high NE frequencies and low NP frequencies. The plot for F2 score in Figure 4.9, is similar to the one for recall, as it is, precisely, a combination of both recall and precision, the first on with a weight two times greater than the latter.

	Precision maximum			**Recall maximum**		
	NP TF	NE TF	Value	NP TF	NE TF	Value
Text 1	1	0	97,30%	9-20	170-200	71,62%
Text 2	0	6	80,24%	17-20	150-200	80,75%
Text 3	0	8	93,48%	13-20	149-200	76,89%
Text 4	0	20-22	81,82%	2-20	157-200	58,93%
Text 5	0	52-53	93,05%	20	185-200	70,91%
Mean	0,2	17,5	89,18%	16,1	181,1	71,82%

Table 4.5: Threshold frequencies and values of precision and recall maxima for the five texts using Approach 2

	F1 score maximum			**F2 score maximum**		
	NP TF	NE TF	Value	NP TF	NE TF	Value
Text 1	1	16-18	72,58%	9-11	170-200	71,04%
Text 2	1	99-101	76,85%	9	150 - 178	74,89%
Text 3	0	149-172	80,66%	0	149-172	77,46%
Text 4	2	157-172	67,35%	2-4	157-172	62,03%
Text 5	1	185-200	77,94%	1	185-200	72,38%
Mean	1	126,9	75,08%	4,6	173,3	71,56%

Table 4.6: Threshold frequencies and values of F1 score and F2 score maxima for the five texts using Approach 2

As in Approach 1, the threshold frequencies (for NE and for NP) for which maxima are obtained for each of the measures, as well as their values, are shown for each one of the five texts considered.

Having a look at Tables 4.5 and 4.6, it could in principle be said that the proposed hypothesis is supported.

The fact that the precision and recall maximums (see Table 4.5) are mostly found at low and high frequency threshold values for both term types, respectively, is no surprise. Values for Text 1 are an exception of the tendency for Precision, as can be appreciated.

F1 and F2 scores (see Table 4.6), however, are optimized when the threshold frequency for NP is set to low values (mid-low in case of NE, for giving recall more importance) and the one for NE, on the other hand, takes high values. This agrees with the hypothesis that noun phrases must have a much smaller threshold frequency than Named Entities in order to be considered sensitive. Thus, by separating the threshold frequency ranges for the two term types, we are allowed to obtain combinations for which both false positive and false negative rates are low. This results in an improvement of the results with respect to the previous approach, as can be seen from the mean maxima values in the tables.

It has to be stated that this section was to be named “Tool validation”. That is, because the project initially took off as an upgrade of the anonymization tool created for CIPSEC, the idea was to validate this upgrade in real sceneries (CIPSEC pilots). A redefinition of the objectives to more ambitious ones, and consequently the creation of a new independent tool for a totally different framework, however, required changing this methodology to computing model performance measures in function of certain parameters and, thus, *evaluating* the tool.

The Python codes used for the evaluation, which are `evaluationscript.py`, `evaluationscriptapproach2.py` and `evaluationfunctions.py`, can be found in Appendix III: Python code files.

5. The anonymization tool

In this section, the anonymization tool v.04 is presented. It should be noted that this version is the last version available at the end of the final project degree period, but many upgrades and improvements could be further studied and implemented, the number of possible versions being indefinite.

In section 3, the process implied in the development of the tool has been commented, and the different techniques and approaches leading to a qualitatively better performance have been discussed. Moreover, the tool performance has been evaluated in section 4 by computing performance measures as a function of the threshold frequencies used. From this, it has been inferred that the best approach to use is the one that considers different levels of sensitivity, or "risk", depending on the word classification, that is, that the frequency threshold for named entities should have greater value than that for noun phrases.

All in all, the development of a Graphical User Interface has been considered necessary to enable an intuitive and user-friendly interaction. Because the tool is intended not only for users with a certain background on the subject but for all types of users, some features have to be further simplified.

An example of this is the frequency thresholds used by the tool when deciding whether a term is sensitive or not. A user is likely not to know what frequency threshold means in this context or to have a notion on whether a certain value is appropriate or not, and explaining the theory involved would be rather tedious and time-consuming. That is why the option of having some pre-defined values for these frequencies has been included. In order to select these frequencies, we have relied on the values obtained for the measures maxima in section 4. Specifically, the ones that should receive most attention are F1 and F2 score. Despite being aware that five is far from a representative number of elements in statistics, in order to have concrete values to part from, we have computed the mean value of the frequency thresholds for the which F1 and F2 scores maxima occur.

	F1 score maximum		**F2 score maximum**	
	NP TF	NE TF	NP TF	NE TF
Mean value	1	126,9	4,6	173,3

Table 5.1: Mean value of the threshold frequencies for the which F1 and F2 score maxima occur.

The reason why both F1 score and F2 score have been considered is because, while the first one takes precision and recall equally into account, the latter gives recall the double of weight than the one given to precision, thus penalizing false negatives more than false positives. The choice of taking account both values permits to define a certain flexibility concerning the privacy-utility trade-off, allowing to select a privacy level ranging from the case in which recall and precision are given equal importance, to the case in the which recall is given twice the importance. Because the aim is to enforce privacy, giving

more weight to precision has not been considered. Bearing this into account, and taking a look at the mean values obtained, five different privacy policies may be defined so that the user may select the one that best suits the need for utility or privacy in the framework in question. That is, the required privacy level in the context of a post in social networks, for example, may be different than the one used for a study in which notes exchanged between different doctors treating a certain patient are analysed. Each level is associated to a tuple of NP frequency-NE frequency as thresholds, so that the user does not have to enter into the underlying theory. These would be, following the simplest approximation:

	NP threshold frequency	**NE threshold frequency**
Level 1	1	100
Level 2	2	125
Level 3	3	150
Level 4	4	175
Level 5	5	200

Table 5.2: NP and NE threshold frequency values associated to each privacy level

Thus, named entities except from persons, that is, organizations, GPEs and facilities, are blacked-out according to the NE threshold frequency. Noun phrases, on the other hand, are first detected as *potentially* sensitive according to the NP threshold frequency, and then it is the user's turn to decide whether they should be considered sensitive or not, then allowing the blacking-out or generalization options if the answer is affirmative.

If the user selects to generalize, the word is searched in WordNet database. If it cannot be found, it is blacked-out. Then, the user must select the appropriate meaning (or s*ynset*), the most adequate hypernym *synset* for that meaning and, finally, the lemma inside this hypernym s*ynset* that best suits the context. If the frequency of the resulting word is still below the one of threshold, the operation is repeated.

A flowchart has been created with the aim of representing the whole process (see Figure 5.1).

Automatic processes are represented in green, while the ones requiring the user interaction are represented in yellow. Also, the specific action executed in a piece of text is coloured in red, and both the input and output texts are represented in blue.

Now, let us see how the anonymization tool is used with the help of the GUI. When executed, a welcome window appears (Figure 5.2), asking to select the text file that one is willing to anonymize.

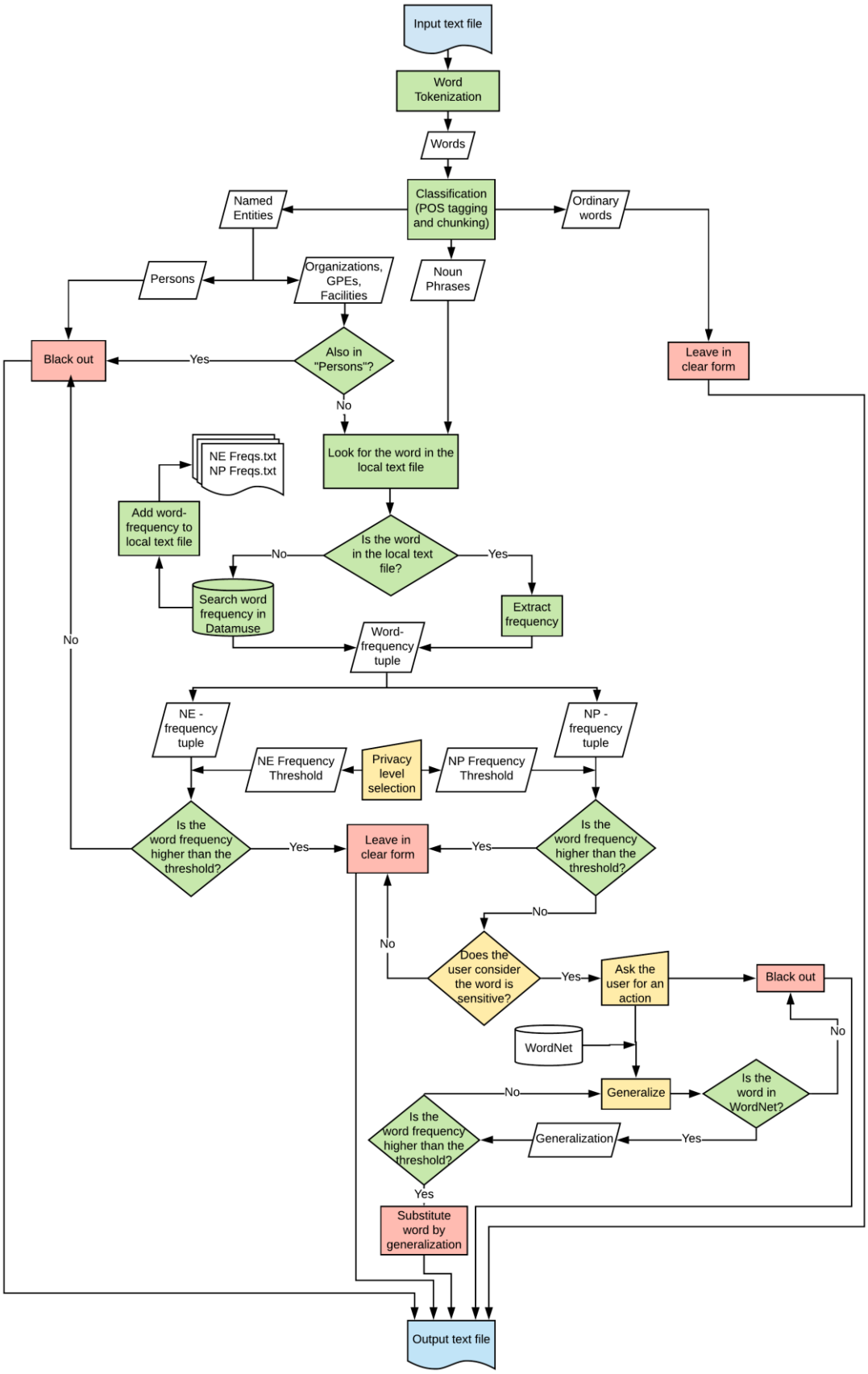

Figure 5.1: Flowchart representing the complete anonymization process

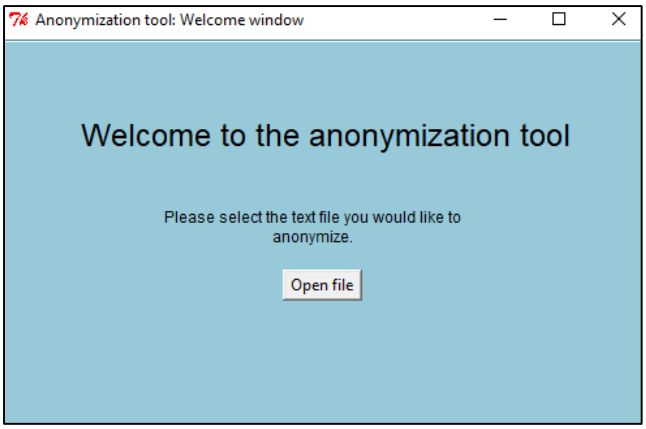

Figure 5.2: Welcome window of the anonymization tool

Once the text is selected, it is processed for anonymization. That is, it is tokenized into words, the noun phrases and named entities are searched and their frequencies are looked for, either in the local dictionary file or by making a query in Datamuse. The user is then asked to select the desired privacy policy (Figure 5.3), that will define the threshold frequencies for NE and for NP.

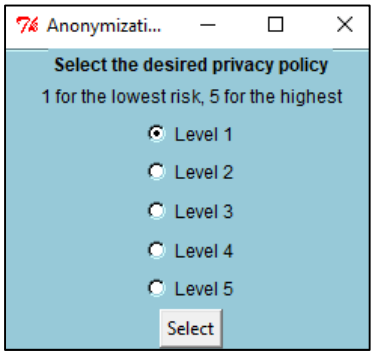

Figure 5.3: Privacy policy selection

Named entities are blacked out regarding their frequency, whereas noun phrases are labelled as potentially sensitive regarding theirs. Thus, the user is asked whether the potentially sensitive noun phrases should be or not obfuscated. The word is shown along with the sentence it appears in, in order to provide some context (Figure 5.4). Note that a potentially sensitive word may appear in more than one sentence and, therefore, a selection must be made for each of the cases.

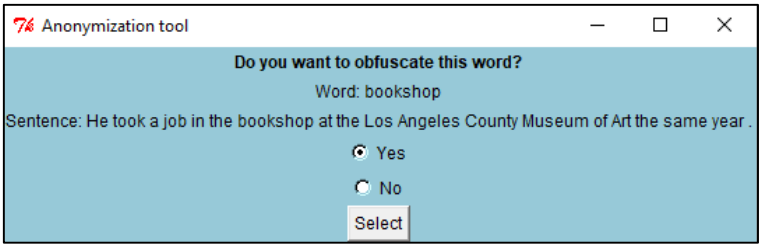

Figure 5.4: Word obfuscation decision

If the user selects “Yes”, he is then asked to choose the desired obfuscation method for that particular term (Figure 5.5): blacking it out or generalizing it, if it is possible (that is, if it is contained in WordNet).

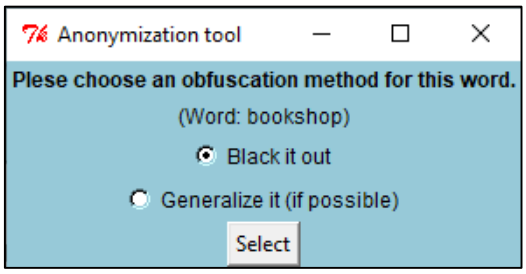

Figure 5.5: Obfuscation method selection

If the user decides to generalize the term, and this one is not present in WordNet, it is alternatively blacked out from the text. If it indeed is found, the user is asked to select the meaning that best adjusts to the term (Figure 5.6) or, in other words, the *synset* it belongs to. The option "None of the others" is also provided.

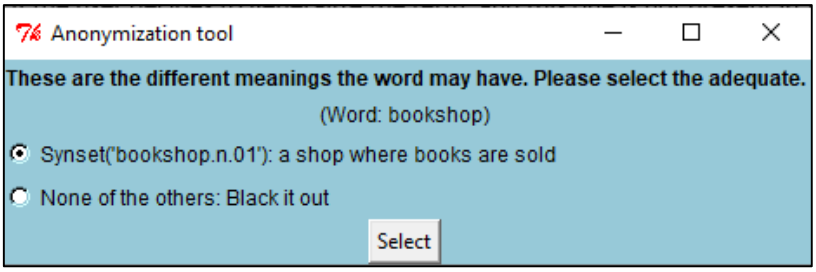

Figure 5.6: Word meaning selection

Once the meaning is selected, we want to look for its generalization. Thus, the hypernyms of the corresponding *synset* are searched. If only one hypernym is found, then that one is automatically selected. If, on the other hand, there is more than one, then the user is asked to decide which one is more appropriate. The same thing happens with the lemmas: once the hypernym has been set, there may be more than one lemma corresponding to that same hypernym *synset*, so the user must decide (Figure 5.7).

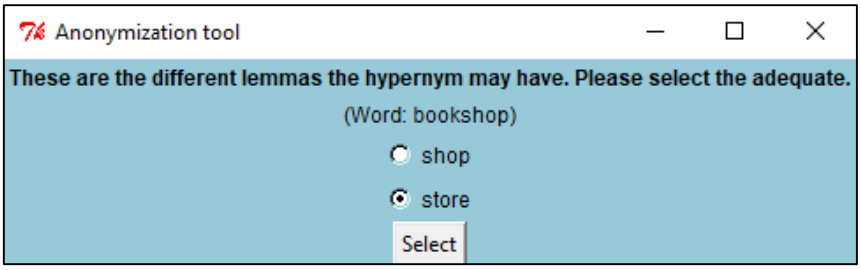

Figure 5.7: Hypernym lemma selection

Once the lemma has been set, its frequency is searched. If it is still below the one of threshold, it has to be generalized further and thus the process we have just seen is repeated. When the frequency value of the generalization is above the one of threshold, the sensitive word is substituted by this one in the output text.

When the interaction process ends, the anonymized text file is obtained and saved.

The code used for the GUI, `final_gui.py`, can be found on Appendix III.

6. Conclusions and further work

The main objective, which was to create an anonymization tool for textual data, has been accomplished.

The initial approach was to make use solely of information theory, that is, to rely only on statistics in order to compute the amount of information revealed by a term, independently from its meaning, as Shannon proposed. That, however, turned out not to work very well in the field of linguistics when put into practice, so making use of Natural Language Processing was considered. Specifically, named entities (NE) and noun phrases (NP) have been detected as potentially sensitive types of words. Named entities tagged as persons have been blacked out from the text, whereas those tagged as organizations, GPEs and facilities have been blacked out or left in clear form regarding their frequency of apparition in a corpus, given by Datamuse API, which then permits to calculate their information content. Thus, if the frequency of apparition is below the one defined as threshold, the term is blacked out. In the case of noun phrases, on the other hand, the terms considered *potentially* sensitive are also found with this approach, but it is then the user who decides whether they actually are sensitive or not. This helps to solve the problem of ambiguity. Moreover, the user gets to determine whether a term should be obfuscated by blacking it out or by generalizing it so that its frequency turns to be above the threshold. In the generalization process, the user gets to make decisions such as the exact meaning of the term in the context, or the generalization that best suits. This improves the semantics conservation of the document and, thus, the utility. A GUI has been developed to ease and improve the experience when interacting with the tool.

An evaluation of the tool has been carried out by doing a word-by-word comparison between five different biographies anonymized by the tool and their by-hand anonymized versions, enabling to compute and discuss several performance measures as a function of the threshold frequencies for NE and NP. This has allowed to choose a set of predefined tuples NP frequency - NE frequency as thresholds, each associated to a certain privacy level that the user selects.

The difficulty regarding anonymization of textual data has been detected, attributed mainly to the enormous complexity, subjectivity and nuances of linguistics. This same reason is the one that causes algorithms such as Named Entity Detectors to incorrectly tag or misinterpret words in certain contexts.

Possible further work could include developing machine-learning anonymization algorithms that learn from the user's decisions, enabling the creation of an efficient automatic anonymization tool for texts. Furthermore, a greater amount of techniques could be included to use combined, such as RegExp, that could identify certain repetition of patterns in the terms labelled as sensitive, and detect terms such as numbers included in dates or identification documents.

To conclude, despite developing a completely automatic tool that performed well would be very difficult because of the problems stated above, further work could be done in terms of using more sophisticated mechanisms that reduced the user's interaction considerably by "learning" from his decisions. That is, to provide the tool with more and more "intelligence" so that decisions are taken automatically and thus the interaction could be greatly reduced in the long run.

References

[1] European Comission. (n.d.). *The General Data Protection Regulation (GDPR)*. Retrieved from https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en

[2] Sweeney, L. (1996). Replacing Personally-Identifying Information in Medical Records, the Scrub System. *Proc. 1996 American Medical Informatics Association*, 333-337.

[3] V. T. Chakaravarthy, H. Gupta, P. Roy, and M. Mohania. (2008). Efficient techniques for document sanitization. *Proc. ACM Conf. Information and Knowledge Management, Vol. 08*, 843–852.

[4] D. Sánchez, M. Batet and A. Viejo. (2013). Automatic general-purpose sanitization of textual documents. *IEEE Transactions on Information Forensics and Security, Vol. 8*, 853-862.

[5] D. Sánchez, M. Batet, and A.Viejo. (2013). Minimizing the disclosure risk of semantic correlations in document sanitization. *Information Sciences, Vol. 249*, 110-123.

[6] D. Sánchez, M. Batet and A. Viejo. (2014). Utility-preserving sanitization of semantically correlated terms in textual documents. *Information Sciences, Vol. 279*, 77-93.

[7] D. Sánchez, A. Viejo. (2016). Enforcing transparent access to private content in social networks by means of automatic sanitization. *Expert Syst., Appl., Vol. 62*, 148–160.

[8] Torra, V. (2017). *Data Privacy: Foundations, New Developments and the Big Data Challenge.* Springer.

[9] Paass, G. (1988). Disclosure risk and disclosure avoidance for microdata. *Journal of Business & Economic Statistics, Vol.6*, 487-500.

[10] Sweeney, L. (2002). k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, Vol. 10*, 557-570.

[11] Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal, Vol. 27*, 379-423.

[12] Shannon, C. E. (1969). Information Theory. In W. Benton, *Encyclopaedia Britannica, Vol. 12* (pp. 246B-249). Chicago.

[13] Stone, J. V. (2018, February 20). Information Theory: A Tutorial Introduction. University of Sheffield, England.

[14] Borda, M. (2011). *Fundamentals in Information Theory and Coding.* Springer.

[15] Kiser, M. (2016, August 11). *Introduction to Natural Language Processing (NLP)*. Retrieved from Algorithmia: https://blog.algorithmia.com/introduction-natural-language-processing-nlp/

[16] Nelson, P. (n.d.). *Search Technologies, Natural Language Processing (NLP) Techniques for Extracting Information.* Retrieved from https://www.searchtechnologies.com/blog/natural-language-processing-techniques

[17] Vallez, M., & Pedraza-Jimenez, R. (2007). *Natural Language Processing in Textual Information Retrieval and Related Topics.* Retrieved from "Hipertext.net": http://www.hipertext.net

[18] Canbek, G., Taskaya Temizel, T., Sagiroglu, S., & Baykal, N. (2017). Binary Classification Performance Measures/Metrics. *2017 International Conference on Computer Science and Engineering (UBMK).* Antalaya, Turkey.

[19] Joshi, R. (2016, September 9). *Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures*. Retrieved from EXSILIO Solutions Blog: https://blog.exsilio.com/

[20] De Ruiter, A. (2015, February 9). *Performance measures in Azure ML: Accuracy, Precision, Recall and F1 Score.* Retrieved from Andreas De Ruiter's BI blog, Microsoft Developer: https://blogs.msdn.microsoft.com/andreasderuiter/

[21] A. Rodríguez-Hoyos, J. A. Estrada-Jimenez, D. Rebollo-Monedero, J. Forné, R. Trapero, A. Alvarez, R. Diaz (2019). *Anonymizing Cybersecurity Data in Critical Infrastructures: The CIPSEC Approach*, Proceedings of the 16th International Conference on Information Systems for Crisis Response And Management, 1198-1208. Track: T13 - Privacy Risk Management in Critical Infrastructure. ISBN: 978-84-09-10498-7. https://iscram2019.webs.upv.es

[22] Python™. Retrieved from: https://www.python.org/

[23] Parikh, M. (2018, October 12). *Advantages Of Python Over Other Programming Languages.* Retrieved from eLearning Industry: https://elearningindustry.com/advantages-of-python-programming-languages

[24] Bird, S., Klein, E., & Loper, E. (2009). Chapter 2: Accessing Text Corpora and Lexical Resources. In *Natural Language Processing with Python:Analyzing Text with the Natural Language Toolkit.* O'Reilly.

[25] *Datamuse API*. (2016, December 5). Retrieved from http://www.datamuse.com/api/

[26] The Google Ngram Viewer Team, part of Google Research. (2013). *Google Books Ngram Viewer*. Recollit de https://books.google.com/ngrams/info

[27] NLTK Project. (2019, June 06). *Source code for nltk.tag.* Retrieved from NLTK 3.4.3 documentation: https://www.nltk.org/_modules/nltk/tag.html

[28] Bird, S., Klein, E., & Loper, E. (2009). Chapter 7: Extracting Information from Text. In *Natural Language Processing with Python:Analyzing Text with the Natural Language Toolkit.* O'Reilly.

[29] Bird, S., Klein, E., & Loper, E. (2009). Chapter 6: Learning to Classify Text. In *Natural Language Processing with Python:Analyzing Text with the Natural Language Toolkit.* O'Reilly.

[30] Johnson, R. M. (2016, May 23). *Lifting the Hood on NLTK's NE Chunker.* Retrieved from Matt's Homepage: https://mattshomepage.com/articles/2016/May/23/nltk_nec/

Appendices

I. Evaluation texts

I.I. Original texts

Text 1

Robert James Adley (2 March 1935 – 13 May 1993) was a Conservative Party politician in the United Kingdom and railway enthusiast . In the 1970s Adley was part-time Marketing Director for Holiday Inn (UK) . He would brief his agency (Alexander James & Dexter) in the morning , before going to the House of Commons . Adley was born Jewish but converted to Anglicanism , [1] and was married with two children . [2] Contents 1 Early life and family 2 Political career 3 Author 4 References 4.1 Citations 4.2 Bibliography 5 External links Early life and family [edit] Robert James Adley born on 2 March 1935 , the son of Harry Adley , a company director . He was educated at Falconbury and Uppingham School , before becoming the Director of Sales at May fair Hotel (1960–64) . In 1961 , he married Jane Elizabeth Pople , daughter of Wilfred Pople of Somerset . Later , he was the Marketing Director at Holiday Inns of Canada . [3] Political career [edit] He was a councillor on Slough Borough Council from 1965 and first stood for Parliament in 1966 for Birkenhead , failing to win the strongly Labour seat . He became member of Parliament for Bristol North East after winning the seat by 462 votes in the 1970 election . Bristol North East was abolished before the next election in 1974 and Adley went on to become member of parliament for Christchurch and Lymington . He would safely hold this seat from 1974 to 1983 , and then after further boundary changes the Christchurch seat from 1983 until his death with one of the largest Conservative majorities in the country . [2] Adley was well known as a railway enthusiast , [4] after gaining a love of trains when he was given The Wonder Book of Trains at the age of three . Adley became leader of the Conservative backbench committee on transport and then the Chairman of the Commons Transport Select Committee . [2] He became a leading opponent of the plans being made by John Major 's government for the privatisation of British Rail , describing it a `` poll tax on wheels `` . [5] Adley had previously opposed the poll tax and bus deregulation , while supporting Concorde and an integrated transport system . Adley also called for talks with the African National Congress and for the UK government to support the aspirations of the black majority in apartheid-era South Africa . [2] Adley died in the Royal Brompton Hospital [6] following a heart attack in 1993 at the age of 58 . [2] After his death the seat was won in a by-election by Liberal Democrat Diana Maddock , but was regained by the Conservatives in 1997 .

Text 2

Dan Amrich (born 5 February 1971) is an American writer , author , actor , musician , and social media expert . He graduated from Ithaca College with a major in Audio

Production and minor in Writing and became a professional journalist and critic with numerous video game and music magazines and websites . He currently works for Ubisoft as a Content Designer , and resides in the San Francisco Bay Area in California . Contents 1 Career 1.1 Journalism 1.2 Social Media 1.3 Trivia 2 Music 3 Writing 4 Personal life 5 Acting 6 References 7 External links Career [edit] Journalism [edit] During the mid-1990s , Amrich wrote for Harris Publications ' Country Guitar Magazine , Guitar World , and SLAM Magazine . He also worked as an editor for Flux Magazine , ran the video games section of America Online ' s Critics ' Choice , and was the Editor in Chief of GameSport Magazine . He also wrote freelance articles for Wired , Time Out New York , and NBA Inside Stuff , among others . In 1996 , Andy Eddy offered him the position of Executive Editor on Digital Diner , an internet lifestyle magazine startup from Metropolis Publications . Only two issues were released before the magazine folded . In 1997 , Amrich began working at GamePro under the editorial aliases Dan Elektro and Bad Hare . He eventually became the features editor , wrote the responses in the reader letters column , and was the main editorial moderator at GamePro.com 's user forums . He also created the GamePro Enigma metapuzzle for the March 2003 issue , [1] and the GamePro Enigma II for the March 2004 issue . Before ending his tenure with the magazine , he authored the book PlayStation 2 For Dummies . [2] From 2003 to 2009 , Amrich worked at numerous Future plc publications . As Senior Editor , he helped launch the US version of GamesRadar in 2005 and appeared in numerous episodes of its podcast TalkRadar , even following his leave of the site . Amrich worked as Senior Editor at Official Xbox Magazine from 2006 to 2009 , where he wrote features , reviews , the letter column , and various other articles for the magazine and its website , OXM Online . [3] He was also the moderator of the OXM discussion forum on Xbox.com . Amrich left Official Xbox Magazine to take on the Editor-in-Chief role for the duration of World of Warcraft : The Official Magazine ' s first quarterly issue . He announced in December 2009 that he would leave journalism to pursue an alternative career path in the video games industry . Before joining Activision , Dan was the weekly co-host of the Official Xbox Magazine podcast alongside Ryan McCaffrey . Social Media [edit] In 2010 , Amrich accepted a position to join Activision as Social Media Manager , [4] with a title change to Community Manager in 2012 until March 2014 . He maintained a community blog , published interviews with video game developers , discussed Activision with its community , and hosted the weekly One of Swords Podcast . He usually alternated between two co-hosts , Katrin Auch [5] and Hugh Sterbakov . [6] Music heard in the One of Swords Podcast was created by Yameen . He usually had one interview per episode from someone working on a current Activision game . In March 2014 , Amrich left Activision [7] to work in the San Francisco Bay area for Ubisoft as the Community Developer for Studio SF . [8] Trivia [edit] Amrich is included as an Easter egg character spry in the arcade basketball game , NBA Hangtime , released in 1996 . Dan interviewed the game 's programmers , Mark Turmell and Sal Divita in 1995 and was invited by the pair to appear as a secret character in the game . Amrich describes the series of events leading to his inclusion into the game , as well as the rationale behind the bizarre bunny rabbit ears he is wearing , on his own website . [9] Dan 's secret character can be unlocked in the arcade , PlayStation , and Nintendo 64 versions by using the code ``

AMRICH `` and PIN `` 2020 `` . [10] Music [edit] Amrich is a vocalist with two years spent in private opera singing training and several years of experience in musical theater . [11] He performed for five years with the San Francisco Bay Area 80s cover band , Fast Times , before leaving to pursue personal musical interests . The result of his departure is the parody band Palette-Swap Ninja , a musical duo featuring Jude Kelley alongside Amrich . Their music comprises modified and humorous covers of popular artists , including Red Hot Chili Peppers , Bowling For Soup , Paul McCartney and more . In addition to vocals , Amrich plays guitar , bass , and on one song , kazoo . [12] in May 2017 , Palette-Swap Ninja released Princess Leia 's Stolen Death Star Plans , a complete album parody of The Beatles ' Sgt . Pepper 's Lonely Hearts Club Band that retells the story of Star Wars in sequential order . [13] [14] [15] The album was released for free in honor of the 40th anniversary of Star Wars and the 50th anniversary of Sgt . Pepper . A series of YouTube videos syncing the music to film footage was created by Katrin Auch . Critical and fan reactions were largely positive ; Star Wars actor Mark Hamill called it `` ingenious `` [16] while Nerdist called it `` the best parody album ever . `` [17] Writing [edit] In February 2012 , Amrich released Critical Path : How To Review Videogames for a Living , which featured a foreword by Gears of War designer , Cliff Bleszinski , who joked that alternate titles could have been `` Abandon All Hope , Ye Who Enter Here `` or `` For the Love of the Game `` . [18] Conceived shortly after Amrich left GamePro Magazine , the book is a guide to videogame journalism which constantly draws on Amrich 's 15 years experience working both sides of the industry to for demonstrations and examples . [19] Self-published , the book was heavily promoted through Laser Time podcast , hosted by former fellow GamesRadar writers and is now available in print and electronically at Amazon.com . Personal life [edit] Amrich is a self-confessed [20] `` fanatic `` of the 80s film franchise , Ghostbusters , and a `` puzzle nerd `` . [21] Dan maintains one of the only sites devoted to Kit Williams and his elaborate , illustrated puzzle book , Masquerade . [22] Acting [edit] Amrich ’ s work as an actor includes commercials and voice-over performance . He has appeared in numerous commercials , [23] and his voice work appears in The Sims : Bustin ' Out . [24] Amrich is also a life member of the International Jugglers ' Association .

Text 3

Colman Robert Hardy Andrews (born February 18 , 1945) is an American writer and editor and authority on food and wine . In culinary circles , he is best known for his association with Saveur magazine , which he founded with Dorothy Kalins , Michael Grossman , and Christopher Hirsheimer in 1994 and where he served as editor-in-chief from 2001 until 2006 . [1] After resigning from the magazine in 2006 , he became the restaurant columnist for Gourmet . [2] In 2010 , he helped launch a food and drink website , The Daily Meal , and served as its editorial director until mid-2018 . He is now a senior editor specializing in food and travel for 24/7 Wall St . He is considered one of the world 's foremost experts on Spanish cuisine , particularly that of the Catalonia region . [3] Contents 1 Early life 2 Early career 3 Magazine life 4 Saveur and Later 5 References 6 External links Early life [edit] Born in Santa Monica , California . His father

, Charles Robert Hardy Douglas Andrews , born in Effingham , Kansas , was a newspaperman , pioneering radio soap opera writer , novelist , and screenwriter . Andrews ' mother was Irene Colman (née Bressette) , an actress of French-Canadian descent born in Nashua , New Hampshire . She played a chorus girl in several Gold Diggers movies and had ingenue roles in a number of other movies . Andrews and his sister , Ann Merry Victoria Andrews (two years his junior) and his older half-sister Joy grew up in the West Los Angeles neighborhood of Holmby Hills . The family moved to Ojai , north of Los Angeles , in 1959 , and Andrews attended Villanova Preparatory School in the same town . Early career [edit] After high school , Andrews went on to Loyola University – now Loyola Marymount University – in Los Angeles as an English major . Kicked out of Loyola after one year for ignoring his studies in favor of the campus radio station , Andrews spent the next year-and-a-half working and traveling , living for brief periods in Atlanta and Cambridge , Massachusetts . Returning to Los Angeles , he enrolled at Los Angeles City College in 1965 . He took a job in the bookshop at the Los Angeles County Museum of Art the same year . In 1968 , after a year-and-a-half at Los Angeles City College and a year at California State University at Los Angeles , Andrews was accepted at the University of California at Los Angeles . He graduated in 1969 with degrees in history and philosophy . Andrews ' first restaurant reviewing job was for The Staff , an offshoot of the LA Free Press . In early 1972 , Andrews , a music lover and amateur singer and songwriter , was hired by the publicity department of Atlantic Records and penned press releases for such albums as Bette Midler 's and Jackson Browne 's debuts . Still writing restaurant reviews on the side , he began to seriously study wine and was a fan in particular of the wine writer Roy Brady , who espoused the notion that a wine should be judged not by its reputation or price but instead by what it smells and tastes like . Brady became his mentor in wine matters . [4] Magazine life [edit] Andrews left Atlantic to become the editor of Coast , a Los Angeles-based lifestyle magazine ; he held the position until 1975 . Meanwhile , Andrews continued reviewing records for Creem , where Lester Bangs was his editor , and covering live music in the LA area for The Hollywood Reporter . He also wrote liner notes for numerous albums , receiving a Grammy nomination in 1972 for notes on a special edition of Miles Davis reissues . In 1975 , Lois Dwan , restaurant reviewer for The Los Angeles Times , asked Andrews to substitute for her while she went on vacation . [5] This began his long association with the newspaper ; though not officially on staff , he was alternately a restaurant reviewer and columnist , book reviewer and travel writer , and the editor of the paper 's travel magazine , Traveling in Style . In 1978 , Andrews , was hired as an associate editor at New West magazine , a bi-weekly California publication started by Clay Felker as a parallel to his seminal New York magazine . He was promoted a year later to senior editor . During this time he met Ruth Reichl , then the restaurant columnist for the Northern California edition of New West , who would go on to become the restaurant critic of the New York Times from 1993 to 1999 , and , later , the editor-in-chief of Gourmet magazine . For a period the two were lovers , their relationship chronicled in Reichl 's memoir Comfort Me with Apples (Random House , 2002) . [6] Andrews left New West in 1980 and began writing for Apartment Life , an urbane lifestyle magazine helmed by Dorothy Kalins . A year later that magazine was

transformed into Metropolitan Home . Over the course of that magazine 's first decade . Andrews wrote about restaurants all over the world ; he was the first American reporter to introduce readers to the great French chef Guy Savoy . [7] Perhaps most significantly , Andrews got a contract to write a book on Catalan cuisine based on an article he 'd written for Met Home . Throughout the eighties he spent a great deal of time traveling to Barcelona and vicinity . The resulting book , Catalan Cuisine , published in 1988 and still in print , has become the standard reference book for restaurant kitchens in that region , and is revered by the top local chefs , including the world-renowned Ferran Adria . [8]

Text 4

Eloise Gerry (January 12 , 1885 – 1970) was an influential research scientist whose early 20th century work contributed greatly to the study of southern pine trees and turpentine production . Gerry was the first woman appointed to the professional staff of the U.S. Forest Service at the Forest Products Laboratory , and one of the first women in the United States to specialize in forest products research . [1] Biography [edit] Eloise Gerry was born January 12 , 1885 in Boston , Massachusetts . She received both bachelor 's and master 's degrees from Harvard University 's Radcliffe College , where she specialized in the anatomy of wood and trees and their physiological responses . She was first hired as a research scientist by the U.S. Forest Service in 1910 . On April 28 , 1914 she addressed the members of the Northern Hemlock and Hardwood Manufacturers Association , at the Forest products laboratory . [2] While working at the new Forest Products Laboratory (FPL) in Madison , Wisconsin , Gerry would also go on to earn a Ph.D. from the University of Wisconsin in 1921 . Her dissertation , based on her research at the FPL , was titled `` Oleoresin Production : A Microscopic Study of the Effects Produced on Woody tissues of Southern Pines by Different Methods of Turpentining . `` Gerry 's research in the area of southern pines and turpentining proved to be her most influential efforts . Working in Mississippi , Gerry performed pioneering work in microscopical studies of the anatomy of resin-yielding pines , and successfully developed methods to increase yield as well as prolong the working life of trees . She worked toward developing best methods stating , `` The microscope reveals many secrets concerning the activities of the tree in producing turpentine and gives these results more quickly than experimental methods alone . `` [3] Based on her field-based research , Gerry was able to develop a program of `` More turpentine , less scar , better pine `` that many later attributed as a savior for the struggling industry . During World War II , Gerry wrote FPL wartime publications on defects in wood used for trainer aircraft and gliders . After the war she worked in the area of research on foreign woods . Following 44 years with the U.S. Forest Service , Gerry retired in 1954 . Gerry died in 1970 at the age of 85 .

Text 5

Myrtle Allen (13 March 1924 – 13 June 2018) was an Irish Michelin star-winning head chef and co-owner of the restaurant The Yeats Room at Ballymaloe House in Shanagarry , County Cork . Besides her career in cooking , she had also been a writer , hotelier and teacher . Contents 1 Life 2 Awards 3 Books 4 References 5 External links Life [edit] Myrtle Hill was the daughter of Henry Houghton Hill , and granddaughter of Arthur Hill , respected architects in Cork . [4] She was a member of the Religious Society of Friends (Quakers) . [5] In 1943 , Myrtle Hill married Ivan Allen , a vegetable grower , who was working at the farm Kinoith in Shanagarry . In 1947 [2] the couple bought Ballymaloe House and the surrounding farm . Ivan managed the fruit and vegetable farm and worked on Kinoith , while his wife took care of the children and the manor . [6] Later , in 1958 , Ivan Allen inherited Kinoith from Wilson Strangman , the deceased owner . As her husband was a successful grower of fruit and vegetables , she had an abundance of fresh products in her kitchen . Under the guidance of her husband , an avid gourmet , she learned to cook by taking cooking courses at the School of Commerce and self-study . By 1962 , she was cookery correspondent of the Irish Farmers Journal . Originally the Irish Farmers Journal was a publication of Macra na Feirme . Myrtle Allen was very active in this young farmers ' organisation , eventually becoming `` Vice President for the Munster Region `` of the `` National Council `` of Macra na Feirme in 1959 . A bid for the presidency in 1963 was unsuccessful . [7] In 1964 , she decided to start a restaurant in her own dining room dubbed The Yeats Room . , [8] as the Allens had several paintings by Jack Yeats . [9] Her philosophy of using local artisanal ingredients and changing her menu daily to reflect the best offerings of the season was `` revolutionary at the time . `` [1] She summed up her philosophy of food in the following nine words " local , seasonal , organic , flavoursome , sustainable and superbly cooked food " . Later she changed a few unused rooms into rooms for a guesthouse , which grew into the hotel Ballymaloe is today . By the 1960s she and her sous-chef , Darina O'Connell , started giving courses in cooking . Later Darina , by then married to Myrtle 's son Tim Allen , moved the cookery classes to Kinoith under the name of Ballymaloe Cookery School . In 1986 Myrtle Allen was part of founding Euro-toques International and founder of Euro-toques Ireland . Euro-toques is an organisation of professional cooks promoting and protecting Europe 's culinary heritage , and defending the quality of local and carefully cooked food . [10] She served as president of the international body from 1994 to 1997 . [11] Myrtle Allen 's husband Ivan died in 1998 . In 2013 Myrtle Allen was the subject of a documentary , [12] Myrtle Allen : A Life in Food , which aired on RTÉ Television . [13] She has been called the `` renowned matriarch of Modern Irish cuisine , `` [1] `` the leading light of modern-day Irish cooking , `` [8] and `` as important to her country 's cuisine as Alice Waters was to America 's . `` [8] Myrtle Allen died 13 June 2018 at Cork University Hospital in Cork after a short battle with pneumonia .

I.II. By-hand anonymized texts

Text 1

*** *** *** (*** *** 1935 – *** *** 1993) was a Conservative Party politician in the United Kingdom and railway enthusiast . In the 1970s *** was part-time Marketing Director for *** *** (UK) . He would brief his agency (*** *** & ***) in the morning , before going to the *** of *** . *** was born Jewish but converted to Anglicanism , [1] and was married with two children . [2] Contents 1 Early life and family 2 Political career 3 Author 4 References 4.1 Citations 4.2 Bibliography 5 External links Early life and family [edit] *** *** *** born on *** *** 1935 , the son of *** *** , a company director . He was educated at *** and *** School , before becoming the Director of Sales at *** *** *** (1960–64) . In 1961 , he married *** *** *** , daughter of *** *** of *** . Later , he was the Marketing Director at *** *** of *** . [3] Political career [edit] He was a councillor on *** *** *** from 1965 and first stood for Parliament in 1966 for *** , failing to win the strongly Labour seat . He became member of Parliament for *** *** *** after winning the seat by *** votes in the 1970 election . *** *** *** was abolished before the next election in 1974 and *** went on to become member of parliament for *** and *** . He would safely hold this seat from 1974 to 1983 , and then after further boundary changes the *** seat from 1983 until his death with one of the largest Conservative majorities in the country . [2] *** was well known as a railway enthusiast , [4] after gaining a love of trains when he was given The Wonder Book of Trains at the age of three . *** became leader of the *** *** *** on *** and then the Chairman of the *** *** *** *** . [2] He became a leading opponent of the plans being made by *** *** 's government for the privatisation of British Rail , describing it a `` *** *** on *** '' . [5] *** had previously opposed the poll tax and bus deregulation , while supporting *** and an integrated transport system . *** also called for talks with the *** National Congress and for the UK government to support the aspirations of the black majority in apartheid-era South Africa . [2] *** died in the Royal *** Hospital [6] following a heart attack in 1993 at the age of 58 . [2] After his death the seat was won in a by-election by Liberal Democrat *** *** , but was regained by the Conservatives in 1997 .

Text 2

*** *** (born *** *** 1971) is an American writer , author , actor , musician , and social media expert . He graduated from *** *** with a major in Audio Production and minor in Writing and became a professional journalist and critic with numerous video game and music magazines and websites . He currently works for *** as a Content Designer , and resides in the *** *** *** *** in California . Contents 1 Career 1.1 Journalism 1.2 Social Media 1.3 Trivia 2 Music 3 Writing 4 Personal life 5 Acting 6 References 7 External links Career [edit] Journalism [edit] During the mid-1990s , *** wrote for *** *** ’ *** *** Magazine , *** *** , and *** Magazine . He also worked as an editor for *** Magazine , ran the video games section of *** *** ’ s *** ’ *** , and was the Editor in Chief of *** Magazine . He also wrote freelance articles for *** , ***

*** *** *** , and *** *** *** , among others . In 1996 , *** *** offered him the position of Executive Editor on *** *** , an internet *** magazine startup from *** Publications . Only two issues were released before the magazine folded . In 1997 , *** began working at *** under the editorial aliases *** *** and *** *** . He eventually became the features editor , wrote the responses in the reader letters column , and was the main editorial moderator at *** 's user forums . He also created the *** *** *** for the March 2003 issue , [1] and the *** *** *** for the March 2004 issue . Before ending his tenure with the magazine , he authored the book *** *** *** *** . [2] From 2003 to 2009 , *** worked at numerous *** *** publications . As Senior Editor , he helped launch the US version of *** in 2005 and appeared in numerous episodes of its *** *** , even following his leave of the site . *** worked as Senior Editor at *** *** Magazine from 2006 to 2009 , where he wrote features , reviews , the letter column , and various other articles for the magazine and its website , *** *** . [3] He was also the moderator of the *** discussion forum on *** . *** left *** *** Magazine to take on the Editor-in-Chief role for the duration of *** of *** : The *** Magazine ' s first quarterly issue . He announced in December 2009 that he would leave journalism to pursue an alternative career path in the video games industry . Before joining *** , *** was the weekly co-host of the *** *** Magazine *** alongside *** *** . Social Media [edit] In 2010 , *** accepted a position to join *** as Social Media Manager , [4] with a title change to Community Manager in 2012 until March 2014 . He maintained a community blog , published interviews with video game developers , discussed *** with its community , and hosted the weekly *** of *** *** . He usually alternated between two co-hosts , *** *** [5] and *** *** . [6] Music heard in the *** of *** *** was created by *** . He usually had one interview per episode from someone working on a current *** *** . In March 2014 , *** left *** [7] to work in the *** *** *** area for *** as the Community Developer for *** *** . [8] Trivia [edit] *** is included as an *** *** character spry in the arcade *** game , *** *** , released in 1996 . *** interviewed the game 's programmers , *** *** and *** *** in 1995 and was invited by the pair to appear as a secret character in the game . *** describes the series of events leading to his inclusion into the game , as well as the rationale behind the *** *** *** *** he is wearing , on his own website . [9] *** 's secret character can be unlocked in the arcade , *** , and *** *** versions by using the code `` *** '' and PIN `` *** '' . [10] Music [edit] *** is a vocalist with two years spent in private opera singing training and several years of experience in musical theater . [11] He performed for five years with the *** *** *** *** 80s cover band , *** *** , before leaving to pursue personal musical interests . The result of his departure is the *** band *** *** , a musical duo featuring *** *** alongside *** . Their music comprises modified and *** covers of popular artists , including *** *** *** *** , *** *** *** , *** *** and more . In addition to vocals , *** plays guitar , bass , and on one song , *** . [12] in May 2017 , *** *** released *** *** 's *** *** *** *** , a complete album *** of The *** ' *** . *** 's *** *** *** *** that retells the story of *** *** in sequential order . [13] [14] [15] The album was released for free in honor of the 40th anniversary of *** *** and the 50th anniversary of *** . *** . A series of YouTube videos syncing the music to film footage was created by *** *** . Critical and fan reactions were largely positive ; *** *** actor *** *** called it ``

ingenious '' [16] while *** called it `` the best *** album ever . `` [17] Writing [edit] In February 2012 , *** released *** *** : *** To *** *** for a *** , which featured a foreword by *** of *** designer , *** *** , who joked that alternate titles could have been `` Abandon All Hope , Ye Who Enter Here '' or `` For the Love of the Game '' . [18] Conceived shortly after *** left *** Magazine , the book is a guide to videogame journalism which constantly draws on *** 's 15 years experience working both sides of the industry to for demonstrations and examples . [19] Self-published , the book was heavily promoted through *** *** *** , hosted by former fellow *** writers and is now available in print and electronically at Amazon.com . Personal life [edit] *** is a self-confessed [20] `` fanatic '' of the 80s film franchise , Ghostbusters , and a `` puzzle nerd '' . [21] *** maintains one of the only sites devoted to *** *** and his elaborate , illustrated puzzle book , *** . [22] Acting [edit] *** ’ s work as an actor includes commercials and voice-over performance . He has appeared in numerous commercials , [23] and his voice work appears in *** *** : *** ' *** . [24] Amrich is also a life member of the *** *** ' *** .

Text 3

*** *** *** *** (born *** *** , 1945) is an American writer and editor and authority on food and wine . In culinary circles , he is best known for his association with *** magazine , which he founded with *** *** , *** *** , and *** *** in 1994 and where he served as editor-in-chief from 2001 until 2006 . [1] After resigning from the magazine in 2006 , he became the restaurant columnist for *** . [2] In 2010 , he helped launch a food and drink website , *** *** *** , and served as its editorial director until mid-2018 . He is now a senior editor specializing in food and travel for *** *** *** . He is considered one of the world 's foremost experts on *** cuisine , particularly that of the *** region . [3] Contents 1 Early life 2 Early career 3 Magazine life 4 *** and Later 5 References 6 External links Early life [edit] Born in *** *** , California . His father , *** *** *** *** *** , born in *** , Kansas , was a newspaperman , pioneering radio soap opera writer , novelist , and screenwriter . *** ' mother was *** *** (neé ***) , an actress of French-Canadian descent born in *** , New Hampshire . She played a *** girl in several *** *** movies and had ingenue roles in a number of other movies . *** and his sister , *** *** *** *** (two years his junior) and his older half-sister *** grew up in the *** *** *** neighborhood of *** *** . The family moved to *** , north of Los Angeles , in 1959 , and *** attended *** *** *** in the same town . Early career [edit] After high school , *** went on to *** University – now *** *** University – in *** *** as an English major . Kicked out of *** after one year for ignoring his studies in favor of the campus radio station , *** spent the next year-and-a-half working and traveling , living for brief periods in *** and *** , Massachusetts . Returning to Los Angeles , he enrolled at *** *** *** College in 1965 . He took a job in the bookshop at the *** *** *** Museum of Art the same year . In 1968 , after a year-and-a-half at *** *** *** College and a year at *** *** University at Los Angeles , Andrews was accepted at the University of *** at *** *** . He graduated in 1969 with degrees in history and philosophy . *** ' first restaurant reviewing job was for The *** , an offshoot of the ***

*** *** . In early 1972 , *** , a music lover and amateur singer and songwriter , was hired by the publicity department of *** *** and penned press releases for such albums as *** *** 's and *** *** 's debuts . Still writing restaurant reviews on the side , he began to seriously study wine and was a fan in particular of the wine writer *** *** , who espoused the notion that a wine should be judged not by its reputation or price but instead by what it smells and tastes like . *** became his mentor in wine matters . [4] Magazine life [edit] *** left *** to become the editor of *** , a *** ***-based lifestyle magazine ; he held the position until 1975 . Meanwhile , *** continued reviewing records for *** , where *** *** was his editor , and covering live music in the LA area for The *** *** . He also wrote liner notes for numerous albums , receiving a *** *** in 1972 for notes on a special edition of *** *** reissues . In 1975 , *** *** , restaurant reviewer for *** *** *** *** , asked *** to substitute for her while she went on vacation . [5] This began his long association with the newspaper ; though not officially on staff , he was alternately a restaurant reviewer and columnist , book reviewer and travel writer , and the editor of the paper 's travel magazine , *** in *** . In 1978 , *** , was hired as an associate editor at *** *** magazine , a *** *** publication started by *** *** as a parallel to his seminal *** *** magazine . He was promoted a year later to senior editor . During this time he met *** *** , then the restaurant columnist for the Northern California edition of *** *** , who would go on to become the restaurant critic of the *** *** *** from 1993 to 1999 , and , later , the editor-in-chief of *** magazine . For a period the two were lovers , their relationship chronicled in *** 's *** *** *** *** *** (*** *** , 2002) . [6] *** left *** *** in 1980 and began writing for *** *** , an urbane lifestyle magazine helmed by *** *** . A year later that magazine was transformed into *** *** . Over the course of that magazine 's first decade . *** wrote about restaurants all over the world ; he was the first American reporter to introduce readers to the great French chef *** *** . [7] Perhaps most significantly , *** got a contract to write a book on *** *** based on an article he 'd written for *** *** . Throughout the eighties he spent a great deal of time traveling to *** and vicinity . The resulting book , *** *** , published in 1988 and still in print , has become the standard reference book for restaurant kitchens in that region , and is revered by the top local chefs , including the world-renowned *** *** . [8]

Text 4

*** *** (*** ***, 1885 – 1970) was an influential research scientist whose early 20th century work contributed greatly to the study of southern pine trees and *** production. *** was the first woman appointed to the professional staff of the U.S. *** *** at the *** *** ***, and one of the first women in the United States to specialize in forest products research. [1] Biography[edit] *** *** was born *** ***, 1885 in ***, Massachusetts. She received both bachelor's and master's degrees from *** University's *** College, where she specialized in the anatomy of wood and trees and their physiological responses. She was first hired as a research scientist by the U.S. *** *** in 1910. On *** ***, 1914 she addressed the members of the *** *** and *** *** Association, at the *** *** ***. [2] While working at the new *** *** *** (***) in ***,

Wisconsin, *** would also go on to earn a Ph.D. from the University of *** in 1921. Her dissertation, based on her research at the ***, was titled "*** ***: A Microscopic Study of the Effects Produced on *** *** of *** *** by Different Methods of ***." ***'s research in the area of southern pines and *** proved to be her most influential efforts. Working in Mississippi, *** performed pioneering work in microscopical studies of the anatomy of *** ***, and successfully developed methods to increase *** as well as prolong the working life of trees. She worked toward developing best methods stating, "The microscope reveals many secrets concerning the activities of the tree in producing *** and gives these results more quickly than experimental methods alone."[3] Based on her field-based research, Gerry was able to develop a program of "More ***, less scar, better pine" that many later attributed as a savior for the struggling industry. During World War II, *** wrote *** wartime publications on defects in wood used for trainer aircraft and gliders. After the war she worked in the area of research on foreign woods. Following 44 years with the U.S. *** ***, *** retired in 1954. **** died in 1970 at the age of 85.

Text 5

*** *** (*** *** 1924 – *** *** 2018) was an Irish *** *** head chef and co-owner of the restaurant *** *** *** at *** *** in *** , *** *** . Besides her career in cooking , she had also been a writer , hotelier and teacher . Contents 1 Life 2 Awards 3 Books 4 References 5 External links Life [edit] *** **** was the daughter of *** *** *** , and granddaughter of *** *** , respected architects in *** . [4] She was a member of the *** Society of *** (***) . [5] In 1943 , *** *** married *** *** , a vegetable grower , who was working at the farm *** in *** . In 1947 [2] the couple bought *** House and the surrounding farm . *** managed the fruit and vegetable farm and worked on *** , while his wife took care of the children and the manor . [6] Later , in 1958 , *** *** inherited *** from *** *** , the deceased owner . As her husband was a successful grower of fruit and vegetables , she had an abundance of fresh products in her kitchen . Under the guidance of her husband , an avid gourmet , she learned to cook by taking cooking courses at the School of *** and self-study . By 1962 , she was cookery correspondent of the *** *** *** . Originally the *** *** *** was a publication of *** *** *** . *** *** was very active in this young farmers ' organisation , eventually becoming `` Vice President for the *** *** '' of the `` *** *** '' of *** *** *** in 1959 . A bid for the presidency in 1963 was unsuccessful . [7] In 1964 , she decided to start a restaurant in her own *** *** dubbed *** *** ***. , [8] as the *** had several *** by *** *** . [9] Her philosophy of using local artisanal ingredients and changing her menu daily to reflect the best offerings of the season was `` revolutionary at the time . `` [1] She summed up her philosophy of food in the following nine words “ local , seasonal , organic , flavoursome , sustainable and superbly cooked food ” . Later she changed a few unused rooms into rooms for a guesthouse , which grew into the hotel *** is today . By the 1960s she and her sous-chef , *** *** , started giving courses in cooking . Later *** , by then married to *** 's son *** *** , moved the cookery classes to *** under the name of *** *** School . In 1986 *** *** was part of founding *** *** and founder

of *** *** . *** is an organisation of professional cooks promoting and protecting Europe 's culinary heritage , and defending the quality of local and carefully cooked food . [10] She served as president of the international body from 1994 to 1997 . [11] *** *** 's husband *** died in 1998 . In 2013 *** *** was the subject of a *** , [12] *** *** *** *** *** *** *** *** , which aired on *** *** . [13] She has been called the `` renowned *** of *** *** cuisine , '' [ 1 ] `` the leading light of *** *** cooking , '' [8] and `` as important to her country 's cuisine as *** *** was to America 's . '' [8] *** *** died 13 June 2018 at *** University Hospital in *** after a short battle with pneumonia .

II. Evaluation plots

II.I. Approach 1

Text 1

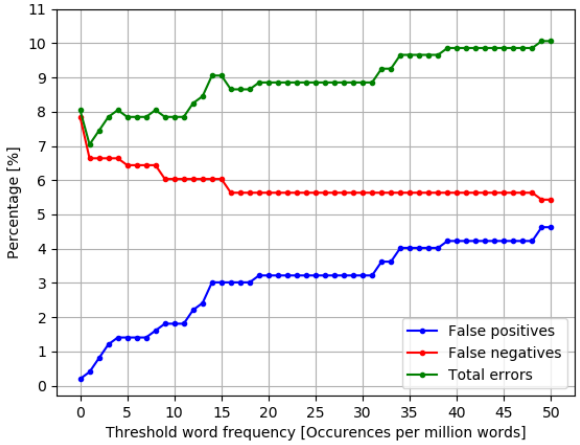

Figure ii.i: Plot of false positives, false negatives and total errors obtained for Text 1 using Approach 1

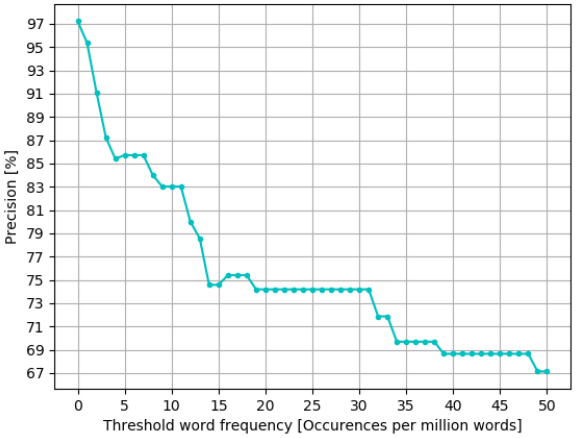

Figure ii.ii: Plot of precision obtained for Text 1 using Approach 1

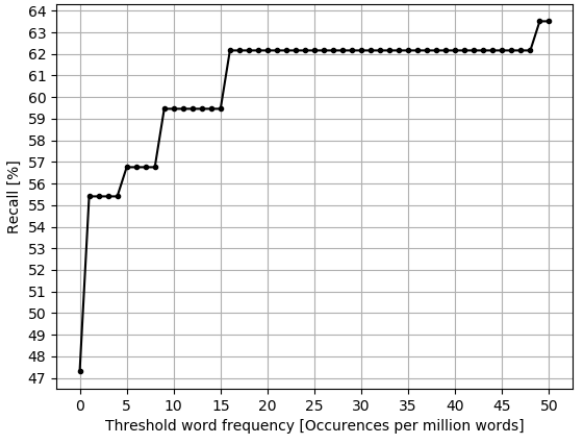

Figure ii.iiii: Plot of recall obtained for Text 1 using Approach 1

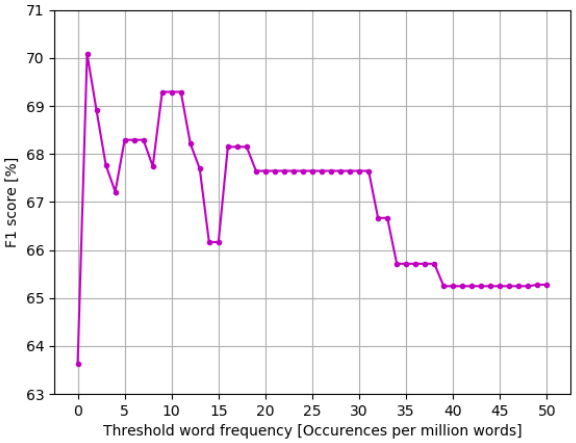

Figure ii.iv: Plot of F1 score obtained for Text 1 using Approach 1

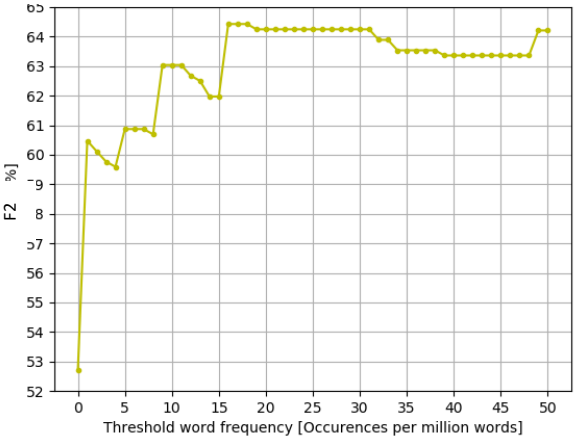

Figure ii.v: : Plot of F2 score obtained for Text 1 using Approach 1

Text 3

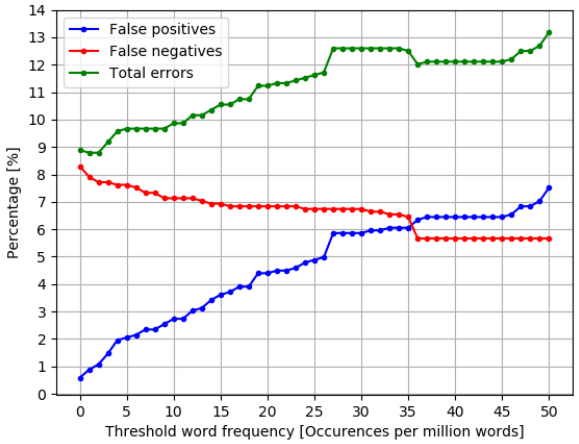

Figure ii.vi: Plot of total errors, false positives and false negatives obtained for Text 3 using Approach 1

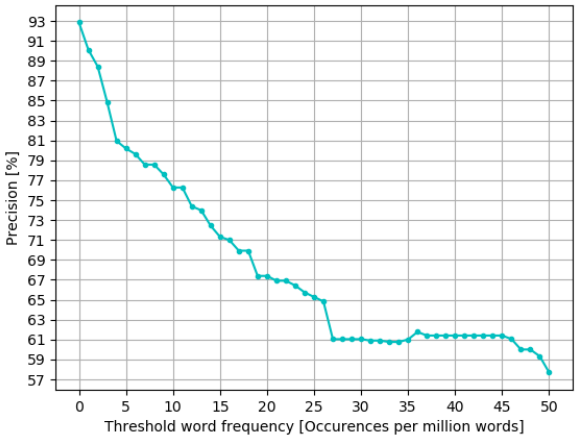

Figure ii.ix: Plot of precision obtained for Text 3 using Approach 1

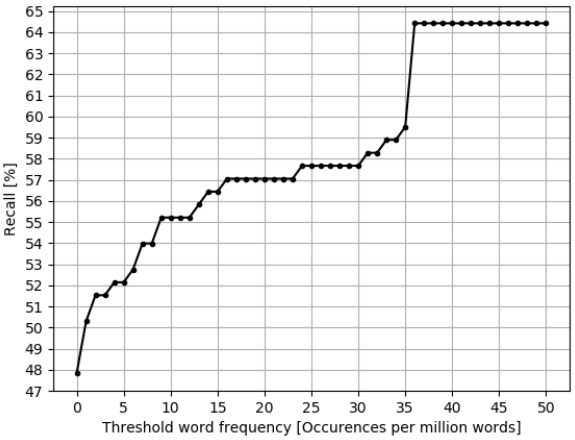

Figure ii.viii: Plot of recall obtained for Text 3 using Approach 1

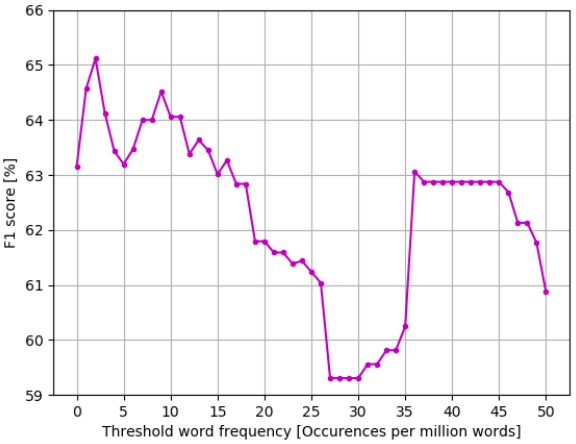

Figure ii.vii: Plot of F1 score obtained for Text 3 using Approach 1

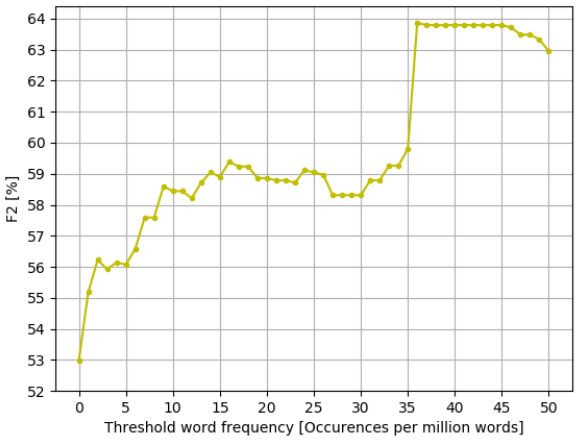

Figure ii.x: Plot of F2 score obtained for Text 3 using Approach 1

Text 4

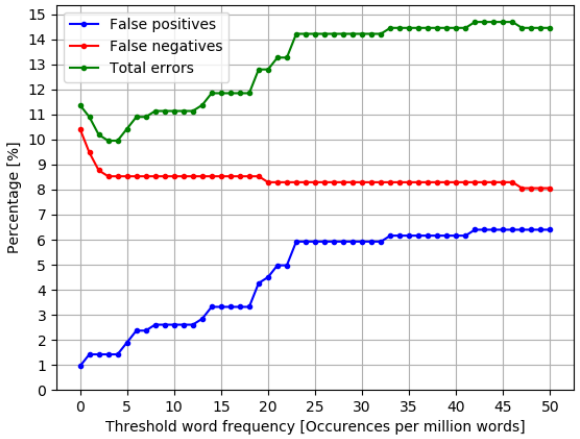

Figure ii.xi: Plot of false positives, false negatives and total errors obtained for Text 4 using Approach 1

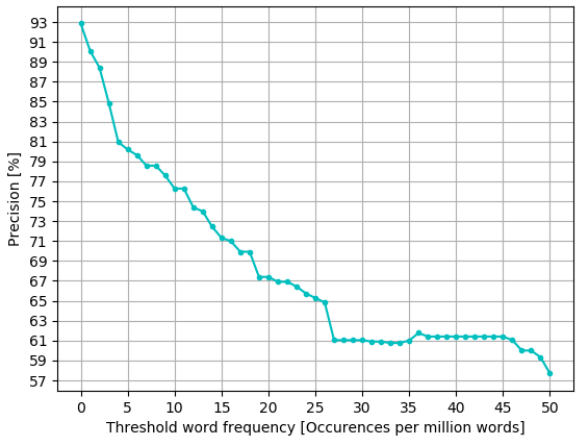

Figure ii.xii: Plot of precision obtained for Text 4 using Approach 1

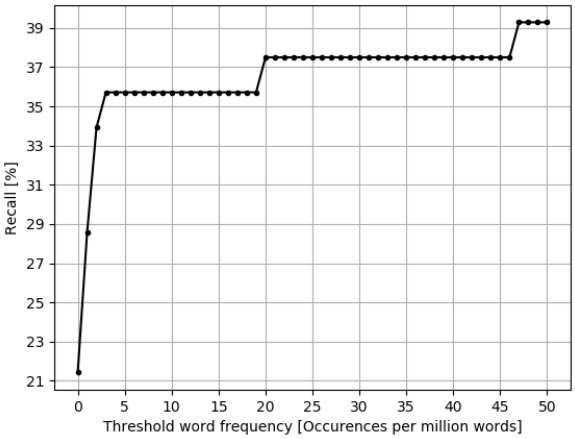

Figure ii.xiii: Plot of recall obtained for Text 4 using Approach 1

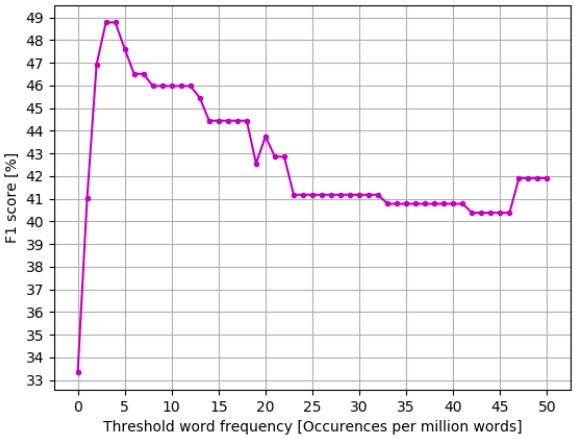

Figure ii.xiv: Plot of F1 score obtained for Text 4 using Approach 1

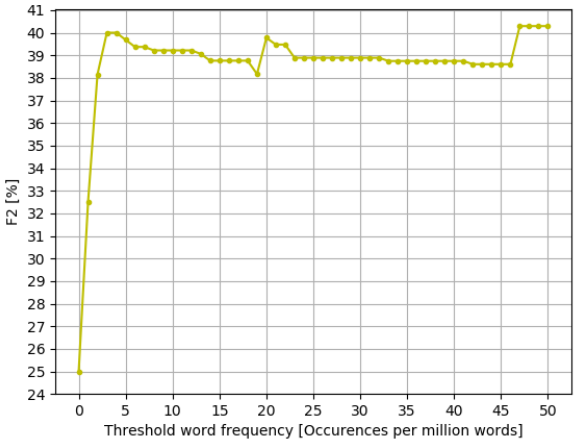

Figure ii.xv: Plot of F2 score obtained for Text 4 using Approach 1

Text 5

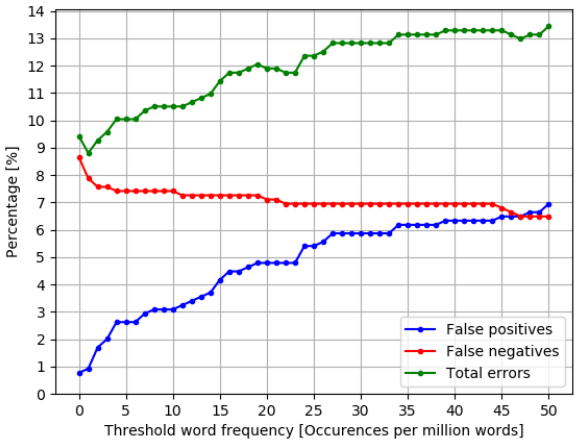

Figure ii.xvi: Plot of false positives, false negatives and total errors obtained for Text 5 using Approach 1

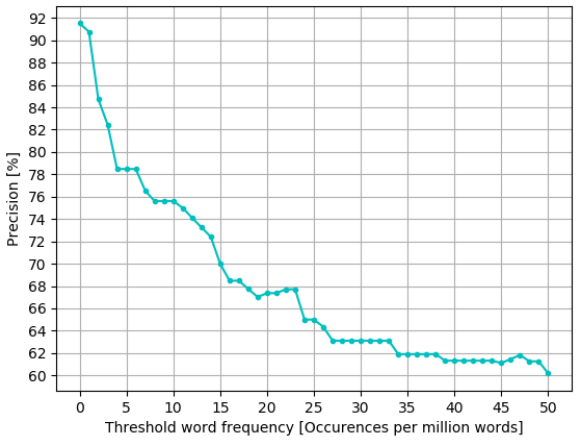

Figure ii.xvii: Plot of precision obtained for Text 5 using Approach 1

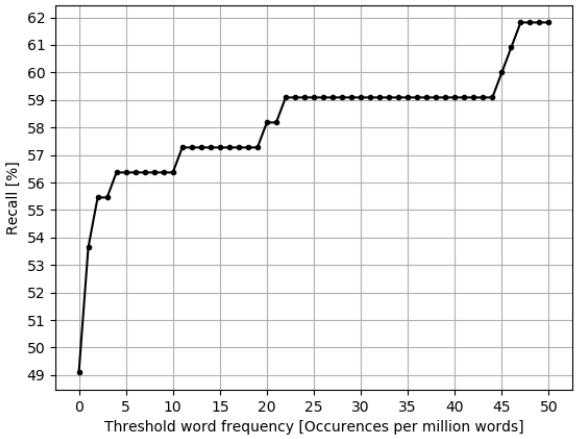

Figure ii.xviii: Plot of recall obtained for Text 5 using Approach 1

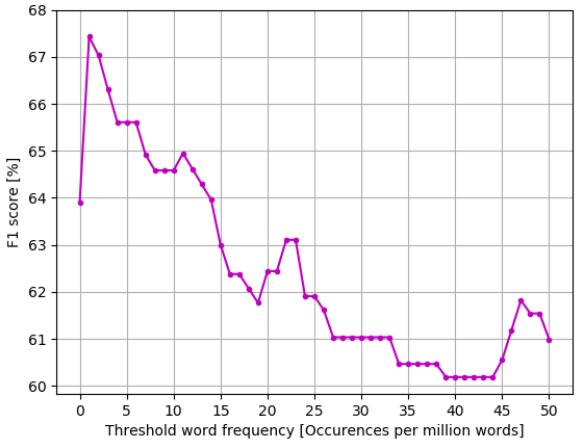

Figure ii.xix*:* Plot of F1 score obtained for Text 5 using Approach 1

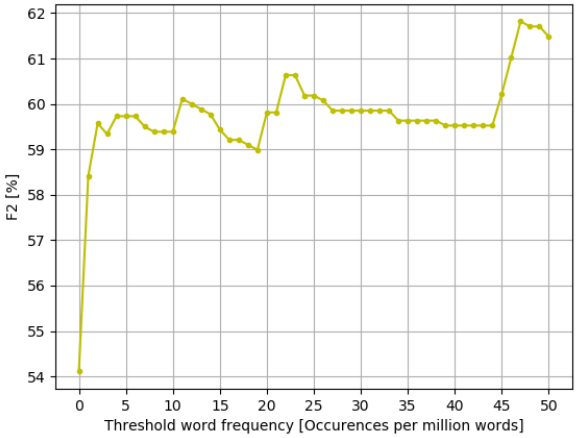

Figure ii.xx: Plot of F2 score obtained for Text 5 using Approach 1

II.II. Approach 2

Text 1

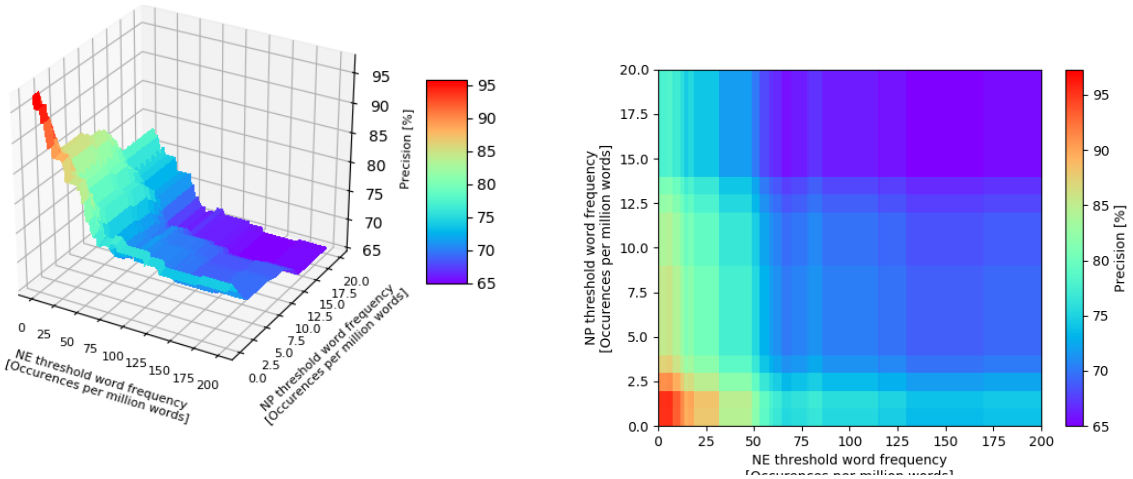

Figure ii.xxi: 3D (left) and 2D (right) surface plots of precision obtained for Text 1 using Approach 2

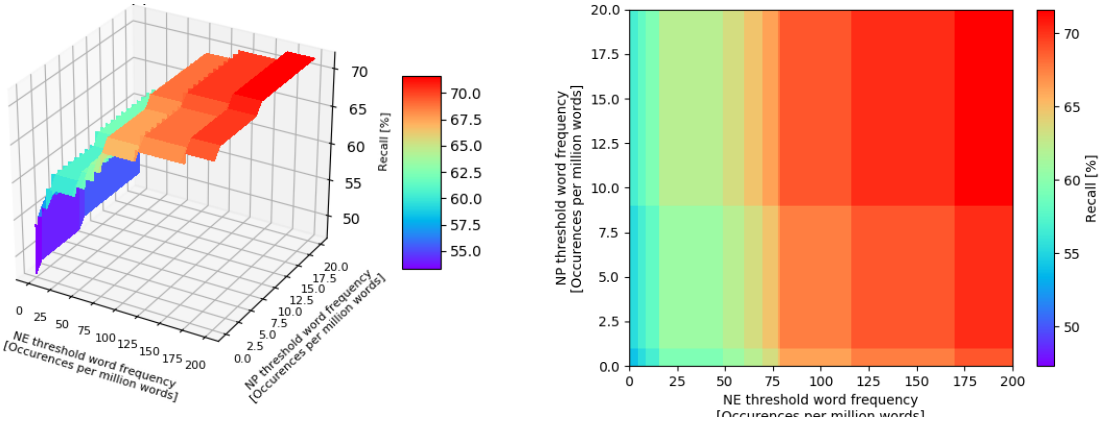

Figure ii.xxii: 3D (left) and 2D (right) surface plots of recall obtained for Text 1 using Approach 2

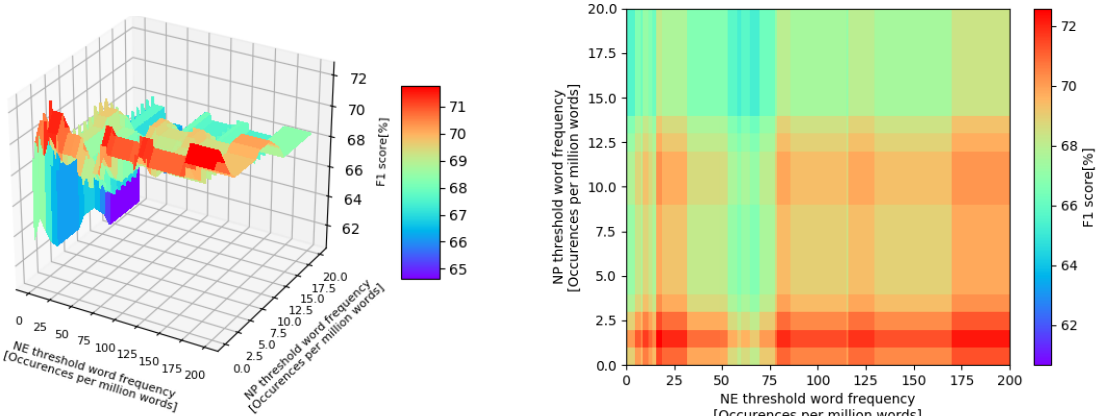

Figure ii.xxiii: 3D (left) and 2D (right) surface plots of F1 score obtained for Text 1 using Approach 2

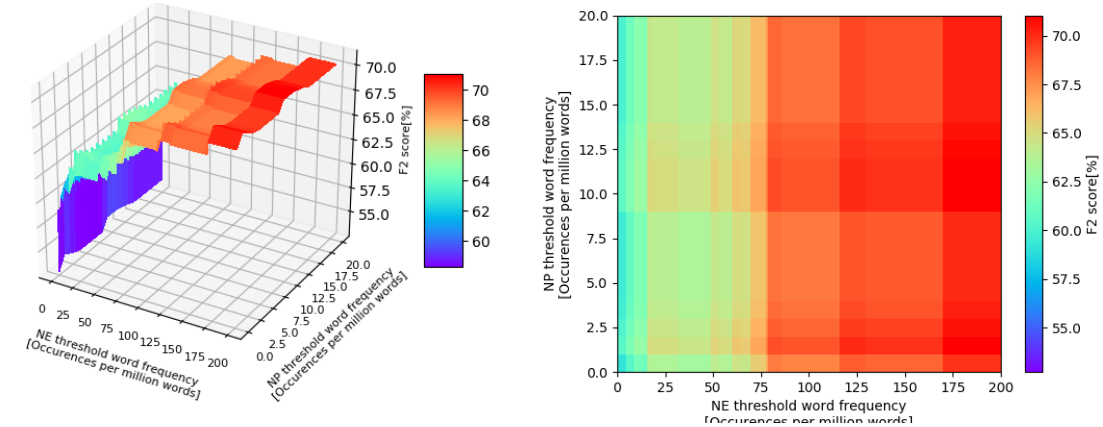

Figure ii.xxiv: 3D (left) and 2D (right) surface plots of F2 score obtained for Text 1 using Approach 2

Text 3

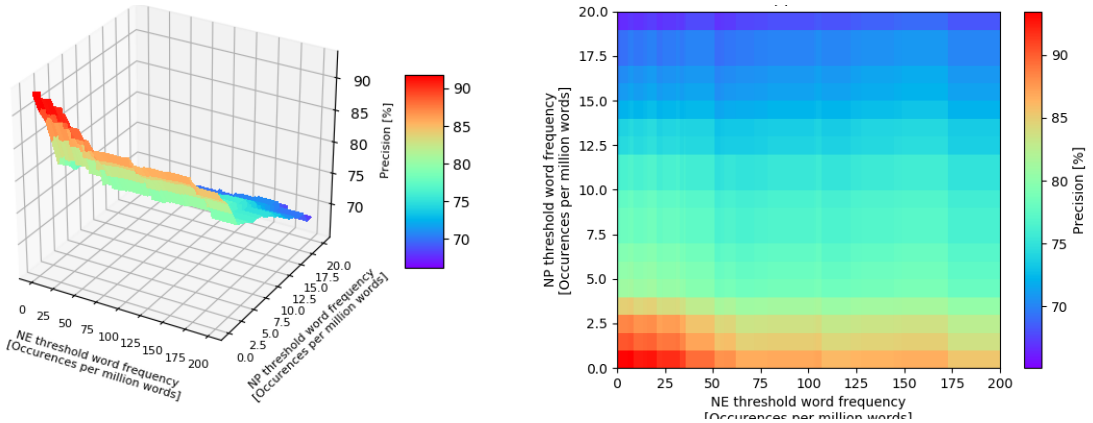

Figure ii.xxv: 3D (left) and 2D (right) surface plots of precision obtained for Text 3 using Approach 2

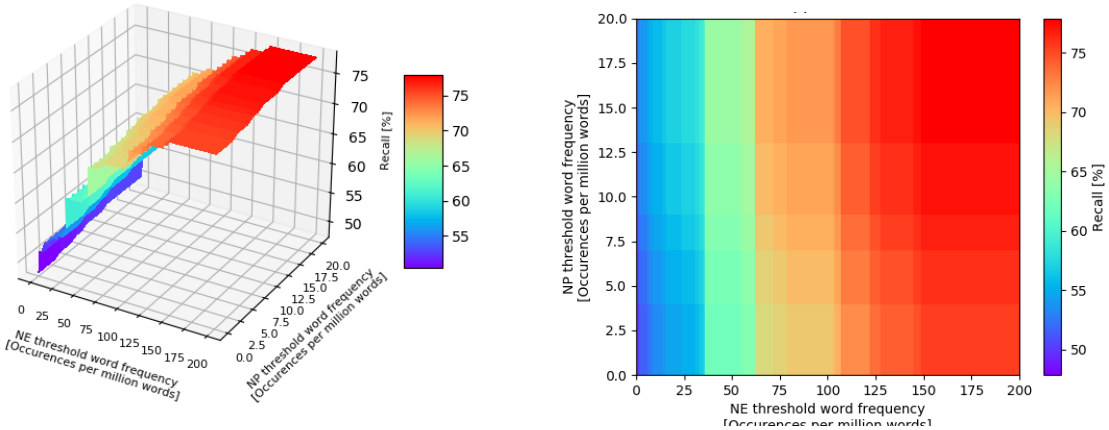

Figure ii.xxvi: 3D (left) and 2D (right) surface plots of recall obtained for Text 3 using Approach 2

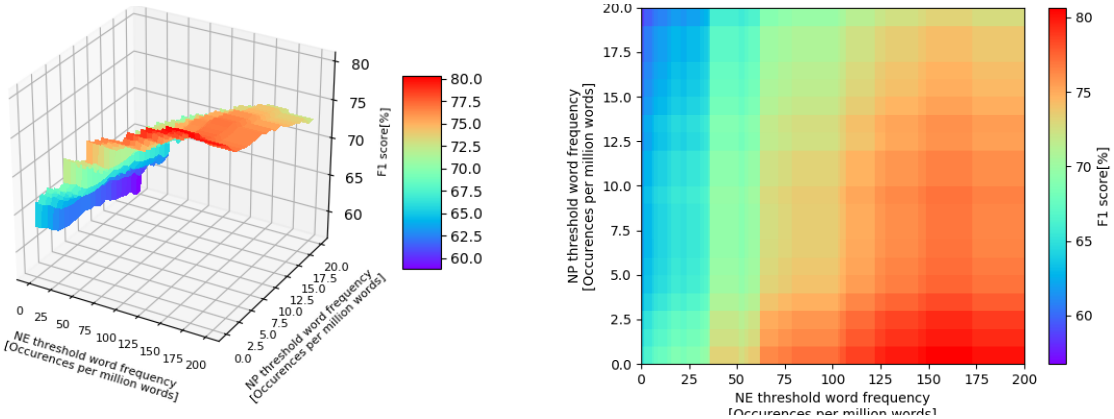

Figure ii.xxvii: 3D (left) and 2D (right) surface plots of F1 score obtained for Text 3 using Approach 2

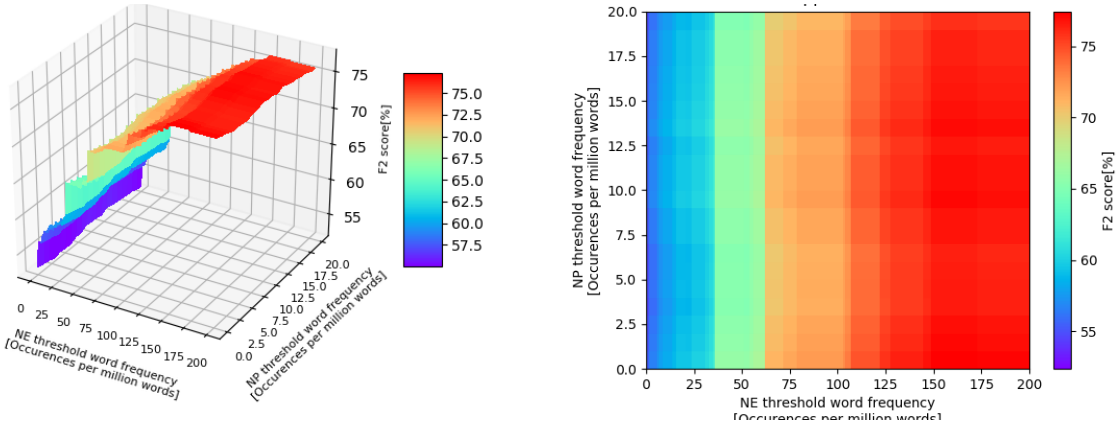

Figure ii.xxviii: 3D (left) and 2D (right) surface plots of F2 score obtained for Text 3 using Approach 2

Text 4

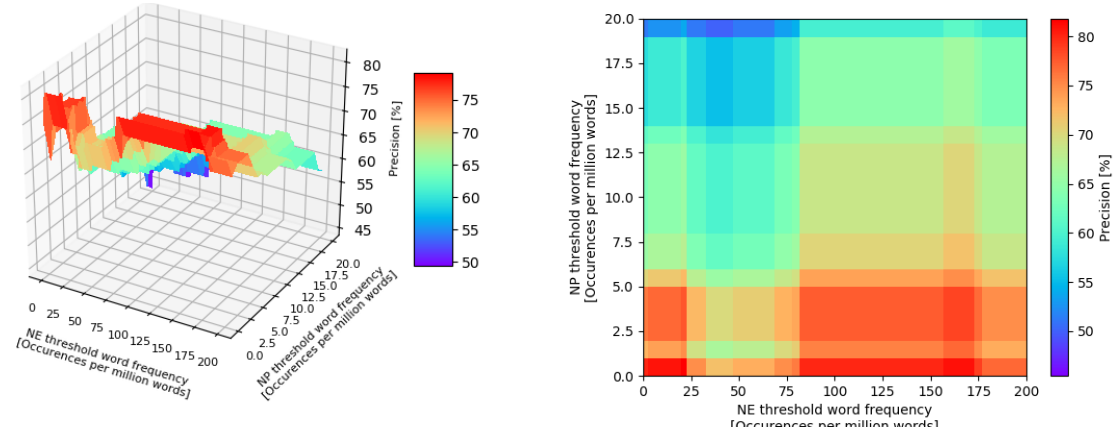

Figure ii.xxix: 3D (left) and 2D (right) surface plots of precision obtained for Text 4 using Approach 2

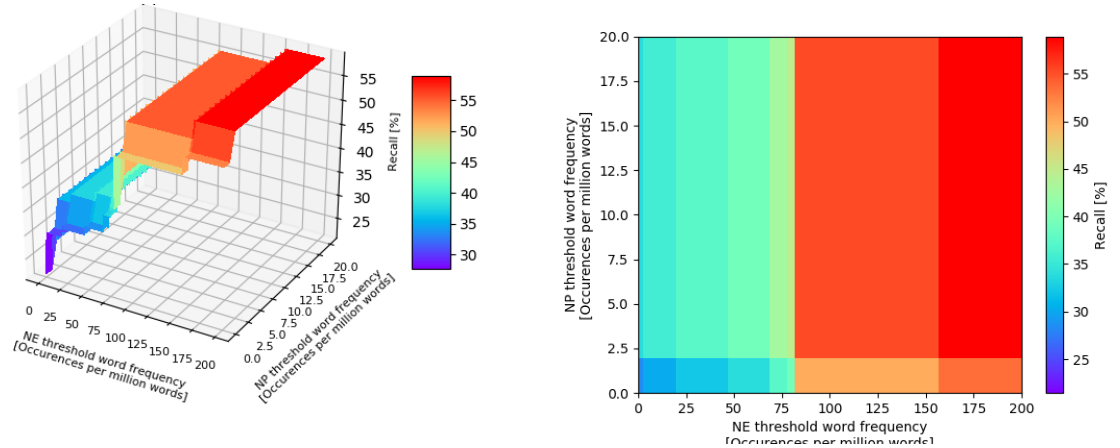

Figure ii.xxx: 3D (left) and 2D (right) surface plots of recall obtained for Text 4 using Approach 2

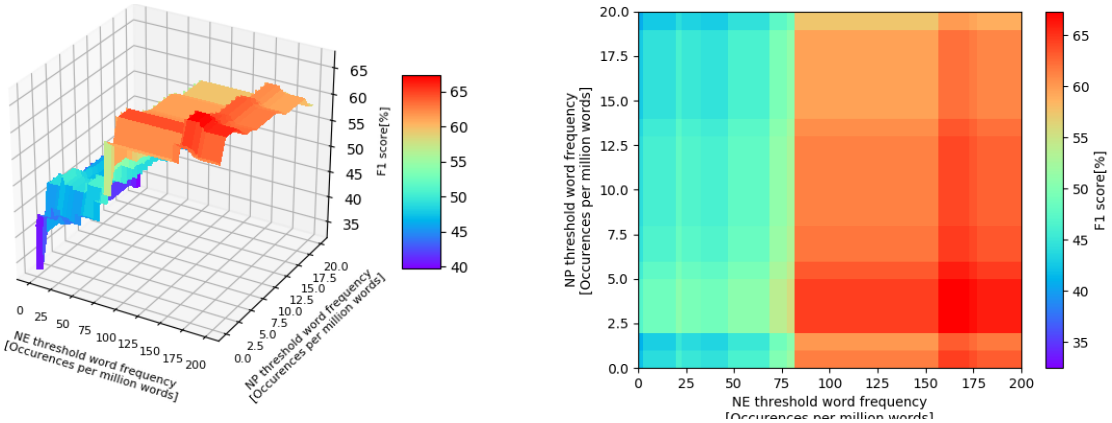

Figure ii. xxxi: 3D (left) and 2D (right) surface plots of F1 score obtained for Text 4 using Approach 2

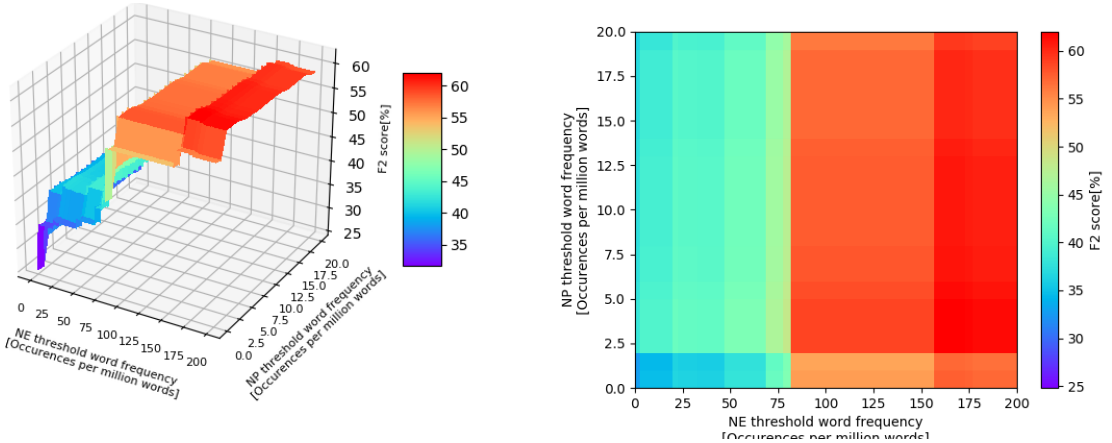

Figure ii. xxxii: 3D (left) and 2D (right) surface plots of F2 score obtained for Text 4 using Approach 2

Text 5

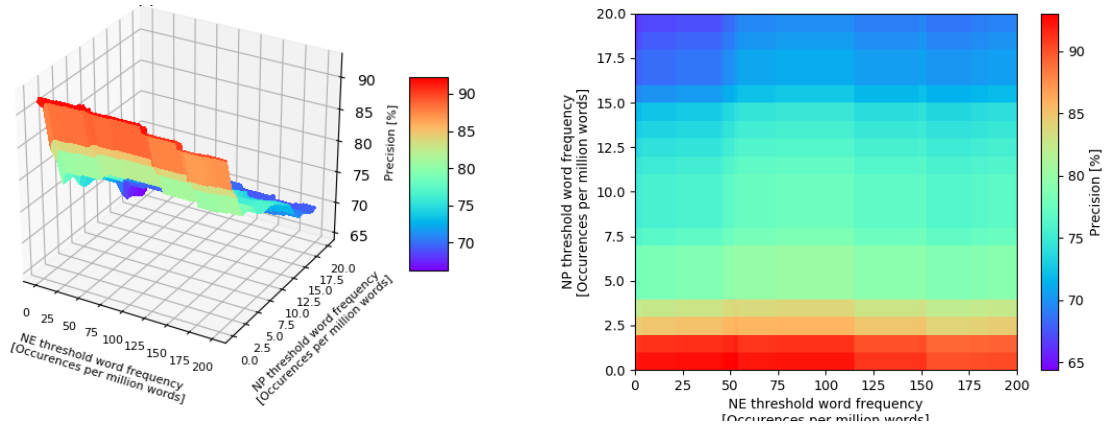

Figure ii.xxxiii: : 3D (left) and 2D (right) surface plots of precision obtained for Text 5 using Approach 2

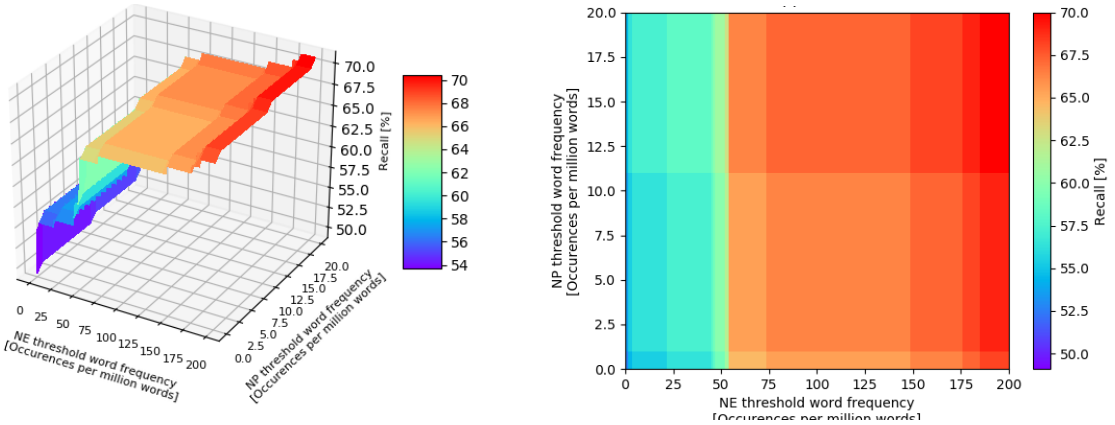

Figure ii.xxxiv: 3D (left) and 2D (right) surface plots of recall obtained for Text 5 using Approach 2

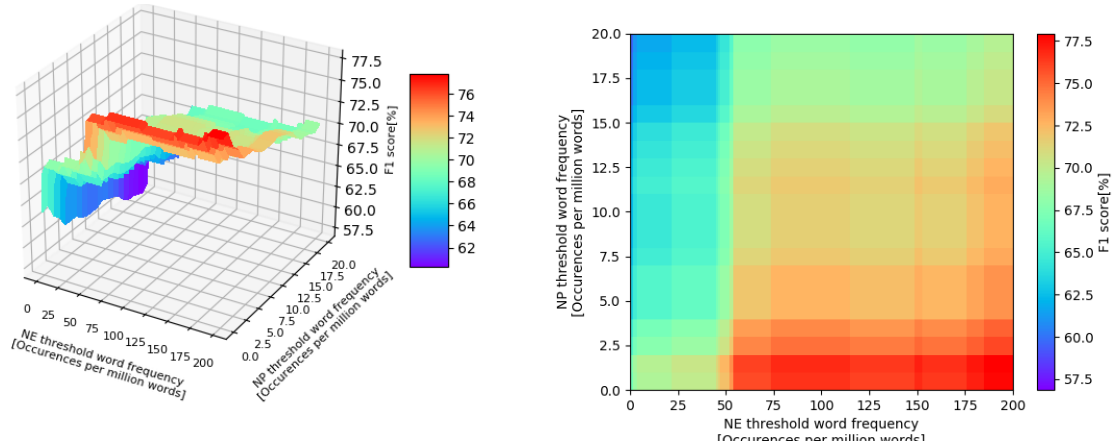

Figure ii.xxxv: 3D (left) and 2D (right) surface plots of F1 score obtained for Text 5 using Approach 2

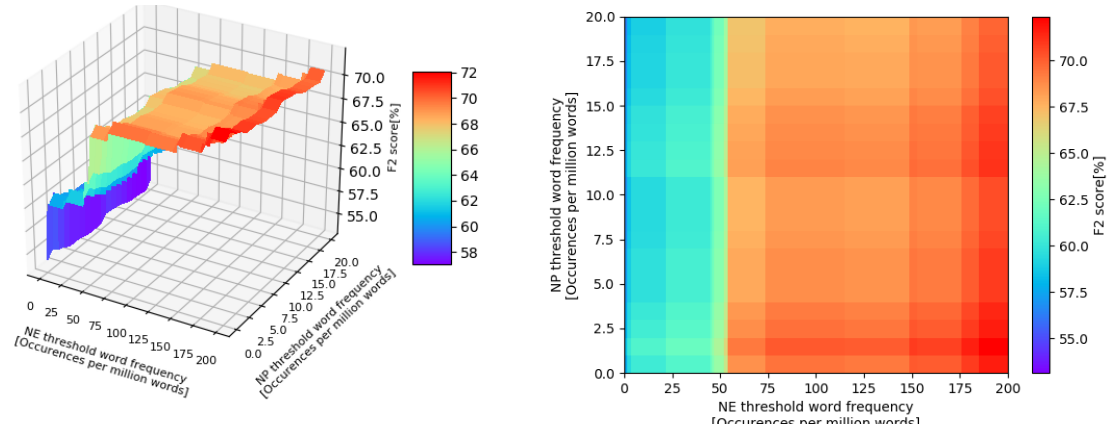

Figure ii.xxxvi: 3D (left) and 2D (right) surface plots of F2 score obtained for Text 5 using Approach 2

III. Python code files

anonym_v04.py

```
import urllib
import urllib2
import nltk
from bs4 import BeautifulSoup
import re
from functionsfile import get_text_from_html , text_to_np_list ,
get_freq, generalize_hyper, get_classified_named_entities ,
get_freqs , get_words_to_anonymize , choose_words_to_anonymize
,wordlist_to_writtable_text
from nltk.corpus import wordnet as wn


text = open("ColmanAndrews.txt")

text = text.read()

text = text.decode("utf-8")

text_words = nltk.word_tokenize(text)

persons, organizations, gpes, facilities =
get_classified_named_entities(text)

ne = organizations + gpes + facilities

list_of_ne = list(set(ne)-(set(persons)))

list_of_np = text_to_np_list(text)

list_of_np = list(set(list_of_np)- set(list_of_ne)- set(persons))

with open("NPFrequencies.txt", "r") as f:

    np_freqs = []

    for line in f.readlines():
        element = line.split()
        word = element[0]
        freq = element[1]
        word = word.decode("utf-8")
        freq = float(freq)
        np_freqs.append([word,freq])

f.close()

with open("NEFrequencies.txt", "r") as f:

    ne_freqs = []

    for line in f.readlines():
        element = line.split()
        word = element[0]
        freq = element[1]
```

```
        word = word.decode("utf-8")
        freq = float(freq)
        ne_freqs.append([word,freq])

f.close()

old_ne = [w for (w,freq) in ne_freqs]

old_np = [w for (w,freq) in np_freqs]

new_np = list(set(list_of_np) - set(old_np))

new_ne = list(set(list_of_ne) - set(old_ne))

new_ne_freqs = get_freqs(new_ne)

new_np_freqs = get_freqs(new_np)

ne_freqs = ne_freqs + new_ne_freqs

np_freqs = np_freqs + new_np_freqs

threshold_freq_ne = 100

threshold_freq_np = 1

ne_to_anonym = get_words_to_anonymize(ne_freqs, threshold_freq_ne)

np_to_anonym = get_words_to_anonymize (np_freqs, threshold_freq_np)

np_to_anonym = list(set(np_to_anonym) - set(ne_to_anonym))

np_hyper = choose_words_to_anonymize(np_to_anonym, text)

np_hyper = generalize_hyper(np_hyper, threshold_freq_np)

words_to_remove_list =  persons + ne_to_anonym

words_to_remove_list = list(set(words_to_remove_list))


words_anonym = ["***" if w in words_to_remove_list
                else w
                for w in text_words]


word_text = []
word_list = []

for (word, gen) in np_hyper:
    indices_in_text = [i for i, w in enumerate(words_anonym) if w
== word]
    indices_in_list = [i for i, w in enumerate(np_hyper) if w[0] ==
word]
    word_text.append([word, indices_in_text])
    word_list.append([word, indices_in_list])
for j in range(len(word_text)):
    item1 = word_text[j]
```

```
    item2 = word_list[j]
    for i in range(len(item1[1])):
        index_text = item1[1][i]
        index_list = item2[1][i]
        words_anonym[index_text] = np_hyper[index_list][1]

text = wordlist_to_writtable_text(text_words)

text_anonym = wordlist_to_writtable_text(words_anonym)

f = open("ColmanAndrews.txt","w")

f.write(text)

f.close()

g = open("ColmanAndrewsAnonym.txt", "w")

g.write(text_anonym)

g.close()
with open("NPFrequencies.txt", "w") as o:

    for (word, freq) in np_freqs:
        word = word.encode("utf-8")
        o.write(str(word) + " " + str(freq) + "\n")

o.close()


with open("NEFrequencies.txt", "w") as f:

    for (word, freq) in ne_freqs:
        word = word.encode("utf-8")
        f.write(str(word) + " " + str(freq) + "\n")

f.close()
```

functionsfile.py

```
from __future__ import division
import urllib
import re
import nltk
from bs4 import BeautifulSoup
from nltk.corpus import wordnet as wn


def get_text_from_html(url):

    html = urllib.urlopen(url).read().decode("utf-8")

    soup = BeautifulSoup(html, features="html.parser")

    # kill all script and style elements
    for script in soup(["script", "style"]):
        script.extract()    # rip it out
```

```
    # get text
    text = soup.get_text()


    return text

def text_to_np_and_propernouns_list(text):

    text_sents = nltk.sent_tokenize(text)

    text_sents = [nltk.word_tokenize(sent) for sent in text_sents]

    text_sents_tagged = [nltk.pos_tag(sent) for sent in text_sents]

    grammar = r"""
        NP : {<JJ>*<NN|NNS>}
             {<NNP|NNPS>+}


    """

    cp = nltk.RegexpParser(grammar)

    list_of_np_tag = []
    list_of_np =[]

    for sent in text_sents_tagged:
        dic = cp.parse(sent)
        print dic
        for item in dic:
            if type(item) is nltk.Tree:
                np = item[0:]
                list_of_np_tag.append(np)
    for lst in list_of_np_tag:
        for (word, tag) in lst:
            list_of_np.append(word)

    return list_of_np

def text_to_np_list(text):

    text_sents = nltk.sent_tokenize(text)

    text_sents = [nltk.word_tokenize(sent) for sent in text_sents]

    text_sents = [nltk.pos_tag(sent) for sent in text_sents]

    grammar = r"""
        NP : {<JJ>*<NN|NNS>}

    """

    cp = nltk.RegexpParser(grammar)

    list_of_np_tag = []
    list_of_np =[]

    for sent in text_sents:
```

```
        dic = cp.parse(sent)
        for item in dic:
            if type(item) is nltk.Tree:
                np = item[0:]
                list_of_np_tag.append(np)
    for lst in list_of_np_tag:
        for (word, tag) in lst:
            list_of_np.append(word)

    return list_of_np

def get_all_named_entities (text):

    text_sents = nltk.sent_tokenize(text)
    text_sents = [nltk.word_tokenize(sent) for sent in text_sents]
    text_sents = [nltk.pos_tag_sents(sent) for sent in text_sents]

    named_entities_tuples = []
    named_entities =[]

    for sent in text_sents:
        tree = nltk.ne_chunk(sent, binary = True)
        for item in tree:
            if type(item) is nltk.Tree:
                ne = item[0:]
                print ne
                named_entities_tuples.append(ne)

    for lst in named_entities_tuples:
        for (word, tag) in lst:
            named_entities.append(word)

    return named_entities

def get_classified_named_entities(text):

    text_sents = nltk.sent_tokenize(text)
    text_sents = [nltk.word_tokenize(sent) for sent in text_sents]
    text_sents = [nltk.pos_tag(sent) for sent in text_sents]

    person_tuples = []
    persons =[]

    organization_tuples = []
    organizations = []

    facility_tuples = []
    facilities =[]

    gpe_tuples =[]
    gpes = []

    for sent in text_sents:

        tree = nltk.ne_chunk(sent)

        for item in tree:
            if type(item) is nltk.Tree:
```

```
                if item.label() == "PERSON":
                    ne = item[0:]
                    person_tuples.append(ne)
                elif item.label() == "ORGANIZATION":
                    ne = item[0:]
                    organization_tuples.append(ne)
                elif item.label() == "GPE":
                    ne = item[0:]
                    gpe_tuples.append(ne)
                elif item.label() == "FACILITY":
                    ne = item[0:]
                    facility_tuples.append(ne)

    for lst in person_tuples:
        for (word, tag) in lst:
            persons.append(word)

    for lst in organization_tuples:
        for (word, tag) in lst:
            organizations.append(word)

    for lst in gpe_tuples:
        for (word, tag) in lst:
            gpes.append(word)

    for lst in facility_tuples:
        for (word, tag) in lst:
            facilities.append(word)

    return persons, organizations, gpes, facilities


def get_freqs (list_of_np):
#from the list of np, returns each word with its frequency in
Datamuse
    words_freqs = []

    for word in list_of_np:
        url = "http://api.datamuse.com/words?sp=xxxxx&md=f&max=1"
        new_url = re.sub("xxxxx" , word , url)
        new_url = new_url.encode("utf-8")
        contents = urllib.urlopen(new_url).read()
        freq = re.findall(r"f:[0-9]+.?[0-9]+", contents)
        if freq == []:
            freq = 0
        else:
            freq = re.findall(r"[0-9]+.?[0-9]+" , freq[0])
            freq = float(freq[0])

        words_freqs.append([word,freq])

    return words_freqs


def get_freq (word):

    url = "http://api.datamuse.com/words?sp=xxxxx&md=f&max=1"
    new_url = re.sub("xxxxx" , word , url)
```

```
    new_url = new_url.encode("utf-8")
    contents = urllib.urlopen(new_url).read()
    freq = re.findall(r"f:[0-9]+.?[0-9]+", contents)
    if freq == []:
        freq = 0
    else:
        freq = re.findall(r"[0-9]+.?[0-9]+" , freq[0])
        freq = float(freq[0])

    return freq

def get_words_to_anonymize (words_freqs, desired_freq):
    words_to_anonym =[]
    for word_freq in words_freqs:
        if 0 < word_freq[1] <= desired_freq :
            words_to_anonym.append(word_freq[0])

    return words_to_anonym



def choose_words_to_anonymize (possible_words, text):

    np_hyper = []

    word_sent =[]

    sents = []

    word_indexes = []

    text_sents = nltk.sent_tokenize(text)
    text_words = [nltk.word_tokenize(sent) for sent in text_sents]

    for word in possible_words:
        for i in range(len(text_words)):
            indices = [j for j, w in enumerate(text_words[i]) if w
== word]
            word_sent = word_sent + ([[word, text_sents[i]]] *
len(indices))


    for (word, sent) in word_sent:


        print "-----------------------------------------------------
--------"
        print "\n"
        print "Word: ", word
        print "\n"
        print "Sentence: " , sent
        print "\n"


        general = "***"

        answer1 = str(raw_input("Do you want to obfuscate this
word? Enter y/n: "))
```

```
        if answer1 == "y":
            answer2 = int(raw_input("""How do you want to obfuscate
it?
            1. Black it out
            2. Generalize it (if possible)

            Enter index: """))

            if answer2 == 1:
                general == "***"
                np_hyper.append([word, general])
            elif answer2 == 2:
                synsets = wn.synsets(str(word))
                if synsets ==[]:
                    print "Sorry, the word could not be found (it
will be blacked out)."
                    general = "***"
                    np_hyper.append([word, general])
                else:
                    if len(synsets) > 1:
                        print "These are the different meanings the
word may have: "
                        for i in range(len(synsets)):
                            print str(i+1) , ": " , synsets[i], ",
" , synsets[i].definition()
                        index = int(raw_input("Which synset does
your word refer to? Enter the index: "))
                        w = synsets[index-1]
                    else:
                        print "There is only one meaning for this
word: ", synsets[0].definition()
                        w = synsets[0]
                    hyps = w.hypernyms()
                    if hyps == []:
                        print "\n"
                        print "Sorry, hypernyms for this word could
not be found (it will be blacked out). "
                        general = "***"
                        np_hyper.append([word, general])
                    else:
                        if len(hyps) > 1:
                            for i in range(len(hyps)):
                                print "\n"
                                print "These are the different
hypernyms the word may have: "
                                print "\n"
                                print str(i+1), ": ", hyps[i]
                            index = int(raw_input("Which hypernym
index does your word refer to? Enter the index: "))
                            lemma_hyp = hyps[index-1].lemmas()
                        else:
                            print "There is only one hypernym for
this word: ", hyps
                            lemma_hyp = hyps[0].lemmas()

                        if len(lemma_hyp) > 1:
                            print "\n"
```

```
                            print "These are the different lemmas
the hypernym may have:"
                            for i in range(len(lemma_hyp)):
                                print str(i+1), ": " , lemma_hyp[i]
                            index = int(raw_input("Which lemma
hypernym does your word refer to? Enter the index: "))
                            general = lemma_hyp[index-1].name()
                            general = re.sub("_", " ", general)
                            general = general.upper()
                            np_hyper.append([word, general])

                        else:
                            general = lemma_hyp[0].name()
                            general = re.sub("_", " ", general)
                            general = general.upper()
                            np_hyper.append([word, general])

        elif answer1 == "n":
            general = word
            np_hyper.append([word,general])



    return np_hyper

def generalize_hyper (np_hyper, threshold_freq):
    for j in range(len(np_hyper)):
        hyper = np_hyper[j][1]
        if hyper.isupper():
            freq = get_freq(hyper)
            general = hyper
            while freq < threshold_freq:
                general = re.sub(" ", "_", general)
                print "\n"
                print "This hypernym needs more generalization: ",
str(general)
                synsets = wn.synsets(str(general))
                if len(synsets) > 1:
                    for i in range(len(synsets)):
                        print str(i+1) , ": " , synsets[i], ", " ,
synsets[i].definition()
                    index = int(raw_input("Which synset does this
word refer to? Enter the index: "))
                    w = synsets[index-1]
                else:
                    print "There is only one meaning for this word:
", synsets[0].definition()
                    w = synsets[0]
                hyps = w.hypernyms()
                if len(hyps) > 1:
                    for i in range(len(hyps)):
                        print "\n"
                        print "These are the different hypernyms
the word may have: "
                        print "\n"
                        print str(i+1), ": ", hyps[i]
                    index = int(raw_input("Which hypernym index
does your word refer to? Enter the index: "))
```

```
                    lemma_hyp = hyps[index-1].lemmas()
                else:
                    print "There is only one hypernym for this
word: ", hyps
                    lemma_hyp = hyps[0].lemmas()
                if len(lemma_hyp) > 1:
                    for i in range(len(lemma_hyp)):
                        print str(i+1), ": " , lemma_hyp[i]
                    index = int(raw_input("Which lemma hypernym
does your word refer to? Enter the index: "))
                    general = lemma_hyp[index-1].name()
                    general = re.sub("_", " ", general)
                    freq = get_freq(general)
                    general = general.upper()
                    np_hyper[j][1] = general
                else:
                    print "There is ony one lemma for this
hypernym: ", lemma_hyp[0]
                    general = lemma_hyp[0].name()
                    general = re.sub("_", " ", general)
                    freq = get_freq(general)
                    general = general.upper()
                    np_hyper[j][1] = general

    return np_hyper

def wordlist_to_writtable_text (list_of_words):
    text = " ". join(list_of_words)
    text = text.encode("utf-8")
    return text
```

evaluationscript.py

```
from __future__ import division
import urllib
import urllib2
import nltk
from bs4 import BeautifulSoup
import re
from functionsfile import get_text_from_html , text_to_np_list ,
get_freq, generalize_hyper, get_classified_named_entities ,
get_freqs , get_words_to_anonymize , choose_words_to_anonymize
,wordlist_to_writtable_text
from evaluationfunctions import test_tool_1, compare_texts,
test_tool_2
from pylab import *
import numpy as np

num = 5

vers = 1

name = "MyrtleAllen"


freqs_ne = np.arange(0, 51, 1)
```

```
text = open(str(name)+".txt")

text = text.read()

text = text.decode("utf-8")

for i in range(len(freqs_ne)):
    test_tool_1(text, freqs_ne[i])

total_positives, total_negatives, false_positives, false_negatives,
total_errors, accuracies, precisions, recalls, f1s, f2s =
compare_texts(freqs_ne)
precmax = max(precisions)

precmaxpos = [i for i, j in enumerate(precisions) if j == precmax]

print "Precisions: " , precmax, "    ", precmaxpos

recmax = max(recalls)

recmaxpos = [i for i, j in enumerate(recalls) if j == recmax]

print "Recall: " , recmax, "    ", recmaxpos

f1max = max(f1s)

f1maxpos = [i for i, j in enumerate(f1s) if j == f1max]

print "F1 score: " , f1max, "    ", f1maxpos

f2max = max(f2s)

f2maxpos = [i for i, j in enumerate(f2s) if j == f2max]

print "F2 score: " , f2max, "    ", f2maxpos

f1 = figure(1)

p11 = plot(freqs_ne, false_positives, "b.-", label = "False
positives")

p12 = plot(freqs_ne, false_negatives, "r.-", label = "False
negatives")

p13 = plot(freqs_ne, total_errors, "g.-", label = "Total errors")

legend()

title("Validation Text "+ str(num) + "\n" + "Version 1")

xlabel("Threshold word frequency [Occurences per million words]")

xticks(np.arange(0, 51.0, 5.0))

minimum = min(false_positives)

maximum = max(total_errors)
```

```
yticks(np.arange(int(minimum), int(maximum)+2))
ylabel("Percentage [%]")
grid()
savefig("ErrorsText"+str(num)+"Version1.png")
f2 = figure(2)
p2 =  plot(freqs_ne, f1s, "m.-" )
title("Validation Text "+str(num)+ "\n" + "Version 1")
minimum = min(f1s)
maximum = max(f1s)
xticks(np.arange( 0.0, 51.0, 5.0))
yticks(np.arange(int(minimum), int(maximum) + 2))
grid()
xlabel("Threshold word frequency [Occurences per million words]")
ylabel("F1 score [%]")
savefig("F1Text"+str(num)+"Version1.png")
f3 = figure(3)
p3 = plot(freqs_ne, accuracies, "y.-")
title("Validation Text "+str(num)+ "\n" + "Version 1")
xticks(np.arange(0.0, 50.0+1, 5.0))
minimum = min(accuracies)
maximum = max(accuracies)
yticks(np.arange(int(minimum)+0.5, int(maximum) + 1 , 0.5))
grid()
xlabel("Threshold word frequency [Occurences per million words]")
ylabel("Accuracy [%]")
savefig("AccuracyText"+str(num)+"Version1.png")
f4 = figure(4)
p4 = plot(freqs_ne, precisions, "c.-")
title("Validation Text "+str(num) + "\n" + "Version 1")
```

```
xticks(np.arange(0, 50.0+1, 5.0))
minimum = min(precisions)
maximum = max(precisions)
yticks(np.arange(int(minimum), int(maximum) + 2 , 2))
grid()
xlabel("Threshold word frequency [Occurences per million words]")
ylabel("Precision [%]")
savefig("PrecisionText"+str(num)+"Version1.png")
f5 = figure(5)
p5 = plot(freqs_ne, recalls, "k.-")
title("Validation Text "+str(num) + "\n" + "Version 1")
xticks(np.arange(0, 50.0+1, 5.0))
minimum = min(recalls)
maximum = max(recalls)
yticks(np.arange(int(minimum),( int(maximum) + 2 ), 2))
grid()
xlabel("Threshold word frequency [Occurences per million words]")
ylabel("Recall [%]")
savefig("RecallText"+str(num)+"Version1.png")
f6 = figure(6)
p6 = plot(freqs_ne, f2s, "y.-")
title("Validation Text "+str(num) + "\n" + "Version 1")
xticks(np.arange(0, 50.0+1, 5.0))
minimum = min(f2s)
maximum = max(f2s)
yticks(np.arange(int(minimum),( int(maximum) + 2 )))
grid()
xlabel("Threshold word frequency [Occurences per million words]")
ylabel("F2 [%]")
```

```
savefig("F2Text"+str(num)+"Version1.png")

show()
```

evaluationscriptapproach2.py

```
from __future__ import division
import urllib
import urllib2
import nltk
from bs4 import BeautifulSoup
import re
from functionsfile import get_text_from_html , text_to_np_list ,
get_freq, generalize_hyper, get_classified_named_entities ,
get_freqs , get_words_to_anonymize , choose_words_to_anonymize
,wordlist_to_writtable_text
from evaluationfunctions import test_tool_1, compare_texts,
test_tool_2, compare_texts_2freqs
from pylab import *
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from pylab import meshgrid, cm, imshow

num = 4
vers = 2
name = "EloiseGerry"

freqs_ne = np.arange(0, 201, 1)

freqs_np = np.arange(0, 21, 1)

text = open(str(name)+".txt")

text = text.read()

text = text.decode("utf-8")

for i in range(len(freqs_ne)):
    for j in range(len(freqs_np)):
        test_tool_2(text, freqs_ne[i], freqs_np[j])

false_positives, false_negatives, total_errors, accuracies,
precisions, recalls, f1s , f2s = compare_texts_2freqs(freqs_ne,
freqs_np)

false_positives = np.array(false_positives)

false_negatives = np.array(false_negatives)

total_errors = np.array(total_errors)

accuracies = np.array(accuracies)
```

```
precisions = np.array(precisions)

recalls = np.array(recalls)

f1s = np.array(f1s)

f2s = np.array(f2s)

max_xy1 = np.where(f1s == f1s.max())

maximumsf1 = zip(max_xy1[0], max_xy1[1])

highestf1 = f1s[maximumsf1[0][0]][maximumsf1[0][1]]

print "F1: " + "\n" + "Positions: " , maximumsf1, "\n" + "Value: "
, highestf1

max_xy2 = np.where(precisions == precisions.max())

maximumsprec = zip(max_xy2[0], max_xy2[1])

highestprec = precisions[maximumsprec[0][0]][maximumsprec[0][1]]

print "Precision: " + "\n" + "Positions: " , maximumsprec, "\n" +
"Value: " , highestprec

max_xy3 = np.where(recalls == recalls.max() )

maximumsrec = zip(max_xy3[0], max_xy3[1])

highestrec = recalls[maximumsrec[0][0]][maximumsrec[0][1]]

print "Recall: " + "\n" + "Positions: " , maximumsrec, "\n" +
"Value: " , highestrec

max_xy4 = np.where(f2s == f2s.max())

maximumsf2 = zip(max_xy4[0], max_xy4[1])

highestf2 = f2s[maximumsf2[0][0]][maximumsf2[0][1]]

print "F2: " + "\n" + "Positions: " , maximumsf2, "\n" + "Value: "
, highestf2


fig = plt.figure(1)

X, Y = np.meshgrid(freqs_ne, freqs_np)

ax = fig.add_subplot(1, 1, 1, projection='3d')

surf = ax.plot_surface(X, Y, false_positives, cmap=cm.rainbow,
                       linewidth=0, antialiased=False)

fig.colorbar(surf, shrink=0.5, aspect=5)
ax.set_xlabel("\n" +'NE threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
```

```
ax.set_ylabel("\n" +'NP threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_zlabel('False positives [%]', fontsize = 8)
ax.set_xticklabels(np.arange(0, 225, 25),fontsize=8)
ax.set_yticklabels(np.arange(0, 22.5, 2.5),fontsize=8)
ax.set_title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("FalsePositivesText"+str(num)+"Version"+str(vers))

fig = plt.figure(2)

X, Y = np.meshgrid(freqs_ne, freqs_np)

ax = fig.add_subplot(1, 1, 1, projection='3d')

surf = ax.plot_surface(X, Y, false_negatives, cmap=cm.rainbow,
                       linewidth=0, antialiased=False)

fig.colorbar(surf, shrink=0.5, aspect=5)

ax.set_xlabel("\n" +'NE threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_ylabel("\n" +'NP threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_zlabel('False negatives [%]', fontsize = 8)
ax.set_title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
ax.set_xticklabels(np.arange(0, 225, 25),fontsize=8)
ax.set_yticklabels(np.arange(0, 22.5, 2.5),fontsize=8)
savefig("FalseNegativesText"+str(num)+"Version"+str(vers))

fig = plt.figure(3)

X, Y = np.meshgrid(freqs_ne, freqs_np)

ax = fig.add_subplot(1, 1, 1, projection='3d')

surf = ax.plot_surface(X, Y, total_errors, cmap=cm.rainbow,
                       linewidth=0, antialiased=False)

fig.colorbar(surf, shrink=0.5, aspect=5)
ax.set_xlabel("\n" +'NE threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_ylabel("\n" +'NP threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_zlabel('Total errors [%]', fontsize = 8)
ax.set_title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
ax.set_xticklabels(np.arange(0, 225, 25),fontsize=8)
ax.set_yticklabels(np.arange(0, 22.5, 2.5),fontsize=8)
savefig("TotalErrorsText"+str(num)+"Version"+str(vers))

fig = plt.figure(4)

X, Y = np.meshgrid(freqs_ne, freqs_np)

ax = fig.add_subplot(1, 1, 1, projection='3d')
```

```
surf = ax.plot_surface(X, Y, accuracies, cmap=cm.rainbow,
                       linewidth=0, antialiased=False)

fig.colorbar(surf, shrink=0.5, aspect=5)

ax.set_xlabel("\n" +'NE threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_ylabel("\n" +'NP threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_zlabel('Accuracy [%]', fontsize = 8)
ax.set_title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
ax.set_xticklabels(np.arange(0, 225, 25),fontsize=8)
ax.set_yticklabels(np.arange(0, 22.5, 2.5),fontsize=8)
savefig("AccuracyText"+str(num)+"Version"+str(vers))

fig = plt.figure(5)

X, Y = np.meshgrid(freqs_ne, freqs_np)

ax = fig.add_subplot(1, 1, 1, projection='3d')

surf = ax.plot_surface(X, Y, precisions, cmap=cm.rainbow,
                       linewidth=0, antialiased=False)

fig.colorbar(surf, shrink=0.5, aspect=5)

ax.set_xlabel("\n" +'NE threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_ylabel("\n" +'NP threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_zlabel('Precision [%]', fontsize = 8)
ax.set_xticklabels(np.arange(0, 225, 25),fontsize=8)
ax.set_yticklabels(np.arange(0, 22.5, 2.5),fontsize=8)
ax.set_title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("PrecisionText"+str(num)+"Version"+str(vers))

fig = plt.figure(6)

X, Y = np.meshgrid(freqs_ne, freqs_np)

ax = fig.add_subplot(1, 1, 1, projection='3d')

surf = ax.plot_surface(X, Y, recalls, cmap=cm.rainbow,
                       linewidth=0, antialiased=False)

fig.colorbar(surf, shrink=0.5, aspect=5)

ax.set_xlabel("\n" +'NE threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_ylabel("\n" +'NP threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_zlabel('Recall [%]', fontsize = 8)
ax.set_xticklabels(np.arange(0, 225, 25),fontsize=8)
ax.set_yticklabels(np.arange(0, 22.5, 2.5),fontsize=8)
ax.set_title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
```

```
savefig("RecallText"+str(num)+"Version"+str(vers))


fig = plt.figure(7)

X, Y = np.meshgrid(freqs_ne, freqs_np)

ax = fig.add_subplot(1, 1, 1, projection='3d')

surf = ax.plot_surface(X, Y, f1s, cmap=cm.rainbow,
                       linewidth=0, antialiased=False)

fig.colorbar(surf, shrink=0.5, aspect=5)

ax.set_xlabel("\n" +'NE threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_ylabel("\n" +'NP threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_zlabel('F1 score[%]', fontsize = 8)
ax.set_xticklabels(np.arange(0, 225, 25),fontsize=8)
ax.set_yticklabels(np.arange(0, 22.5, 2.5),fontsize=8)
ax.set_title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("F1Text"+str(num)+"Version"+str(vers))



fig = figure(8)

X, Y = np.meshgrid(freqs_ne, freqs_np)
im = plt.pcolor(X, Y, false_positives, cmap = plt.cm.rainbow)
cbar = plt.colorbar(im)
cbar.set_label("False positives [%]")
plt.xlabel('NE threshold word frequency' + "\n" + '[Occurences per
million words]')
plt.ylabel('NP threshold word frequency' + "\n" + '[Occurences per
million words]')
plt.title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("FalsePositivesText"+str(num)+"Version"+str(vers)+"surface"
)

fig = figure(9)

X, Y = np.meshgrid(freqs_ne, freqs_np)
im = plt.pcolor(X, Y, false_negatives, cmap = plt.cm.rainbow)
cbar = plt.colorbar(im)
plt.xlabel('NE threshold word frequency' + "\n" + '[Occurences per
million words]')
plt.ylabel('NP threshold word frequency' + "\n" + '[Occurences per
million words]')
cbar.set_label("False negatives [%]")
plt.title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("FalseNegativesText"+str(num)+"Version"+str(vers)+"surface"
)

fig = figure(10)
```

```
X, Y = np.meshgrid(freqs_ne, freqs_np)
im = plt.pcolor(X, Y, total_errors, cmap = plt.cm.rainbow)
cbar = plt.colorbar(im)
plt.xlabel('NE threshold word frequency' + "\n" + '[Occurences per
million words]')
plt.ylabel('NP threshold word frequency' + "\n" + '[Occurences per
million words]')
cbar.set_label("Total errors [%]")
plt.title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("TotalErrorsText"+str(num)+"Version"+str(vers)+"surface")

fig = figure(11)

X, Y = np.meshgrid(freqs_ne, freqs_np)
im = plt.pcolor(X, Y, accuracies, cmap = plt.cm.rainbow)
cbar = plt.colorbar(im)
plt.xlabel('NE threshold word frequency' + "\n" + '[Occurences per
million words]')
plt.ylabel('NP threshold word frequency' + "\n" + '[Occurences per
million words]')
cbar.set_label("Accuracy [%]")
plt.title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("AccuracyText"+str(num)+"Version"+str(vers)+"surface")

fig = figure(12)

X, Y = np.meshgrid(freqs_ne, freqs_np)
im = plt.pcolor(X, Y, precisions, cmap = plt.cm.rainbow)
cbar = plt.colorbar(im)
plt.xlabel('NE threshold word frequency' + "\n" + '[Occurences per
million words]')
plt.ylabel('NP threshold word frequency' + "\n" + '[Occurences per
million words]')
cbar.set_label("Precision [%]")
plt.title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("PrecisionText"+str(num)+"Version"+str(vers)+"surface")

fig = figure(13)

X, Y = np.meshgrid(freqs_ne, freqs_np)
im = plt.pcolor(X, Y, recalls, cmap = plt.cm.rainbow)
cbar = plt.colorbar(im)
plt.xlabel('NE threshold word frequency' + "\n" + '[Occurences per
million words]')
plt.ylabel('NP threshold word frequency' + "\n" + '[Occurences per
million words]')
cbar.set_label("Recall [%]")
plt.title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("RecallText"+str(num)+"Version"+str(vers)+"surface")

fig = figure(14)

X, Y = np.meshgrid(freqs_ne, freqs_np)
```

```
im = plt.pcolor(X, Y, f1s, cmap = plt.cm.rainbow)
cbar = plt.colorbar(im)
plt.xlabel('NE threshold word frequency' + "\n" + '[Occurences per
million words]')
plt.ylabel('NP threshold word frequency' + "\n" + '[Occurences per
million words]')
cbar.set_label("F1 score[%]")
plt.title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("F1Text"+str(num)+"Version"+str(vers)+"surface")


fig = plt.figure(15)

X, Y = np.meshgrid(freqs_ne, freqs_np)

ax = fig.add_subplot(1, 1, 1, projection='3d')

surf = ax.plot_surface(X, Y, f2s, cmap=cm.rainbow,
                       linewidth=0, antialiased=False)

fig.colorbar(surf, shrink=0.5, aspect=5)

ax.set_xlabel("\n" +'NE threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_ylabel("\n" +'NP threshold word frequency' + "\n" +
'[Occurences per million words]', fontsize = 8)
ax.set_zlabel('F2 score[%]', fontsize = 8)
ax.set_xticklabels(np.arange(0, 225, 25),fontsize=8)
ax.set_yticklabels(np.arange(0, 22.5, 2.5),fontsize=8)
ax.set_title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("F2Text"+str(num)+"Version"+str(vers))

fig = figure(16)

X, Y = np.meshgrid(freqs_ne, freqs_np)
im = plt.pcolor(X, Y, f2s, cmap = plt.cm.rainbow)
cbar = plt.colorbar(im)
plt.xlabel('NE threshold word frequency' + "\n" + '[Occurences per
million words]')
plt.ylabel('NP threshold word frequency' + "\n" + '[Occurences per
million words]')
cbar.set_label("F2 score[%]")
plt.title("Validation Text " + str(num) + "\n" + "Approach
"+str(vers))
savefig("F2Text"+str(num)+"Version"+str(vers)+"surface")
plt.show()
```

evaluationfunctions.py

```
from __future__ import division
import urllib
import urllib2
import nltk
from bs4 import BeautifulSoup
import re
```

```
from functionsfile import get_text_from_html ,
text_to_np_and_propernouns_list, text_to_np_list , get_freq,
generalize_hyper, get_classified_named_entities , get_freqs ,
get_words_to_anonymize , choose_words_to_anonymize
,wordlist_to_writtable_text
import numpy as np

def test_tool_1(text, threshold_freq,name):

    text_words = nltk.word_tokenize(text)

    persons, organizations, gpes, facilities =
get_classified_named_entities(text)

    ne = organizations + gpes + facilities

    ne = list(set(ne)- set(persons))

    list_of_ne = list(set(ne))

    list_of_np = text_to_np_list(text)

    list_of_np = list(set(list_of_np) - set(list_of_ne))

    with open("NPFrequencies.txt", "r") as f:

        np_freqs = []

        for line in f.readlines():
            element = line.split()
            word = element[0]
            freq = element[1]
            word = word.decode("utf-8")
            freq = float(freq)
            np_freqs.append([word,freq])

    f.close()

    with open("NEFrequencies.txt", "r") as f:

        ne_freqs = []

        for line in f.readlines():
            element = line.split()
            word = element[0]
            freq = element[1]
            word = word.decode("utf-8")
            freq = float(freq)
            ne_freqs.append([word,freq])

    f.close()

    old_ne = [w for (w,freq) in ne_freqs]

    old_np = [w for (w,freq) in np_freqs]

    new_np = list(set(list_of_np) - set(old_np))
```

```
    new_ne = list(set(list_of_ne) - set(old_ne))

    new_ne_freqs = get_freqs(new_ne)

    new_np_freqs = get_freqs(new_np)

    ne_freqs = ne_freqs + new_ne_freqs

    np_freqs = np_freqs + new_np_freqs

    ne_to_anonym = get_words_to_anonymize(ne_freqs, threshold_freq)

    np_to_anonym = get_words_to_anonymize (ne_freqs,
threshold_freq)

    words_to_remove_list = persons + ne_to_anonym + np_to_anonym

    words_to_remove_list = list(set(words_to_remove_list))

    words_anonym = ["***" if w in words_to_remove_list
                else w
                for w in text_words]

    text = wordlist_to_writtable_text(text_words)

    text_anonym = wordlist_to_writtable_text(words_anonym)

    f = open(str(name)+".txt","w")

    f.write(text)

    f.close()

    g = open(str(name)+"Version1Freq" + str(threshold_freq) +
".txt", "w")

    g.write(text_anonym)

    g.close()

    with open("NPFrequencies.txt", "w") as o:

        for (word, freq) in np_freqs:
            word = word.encode("utf-8")
            o.write(str(word) + " " + str(freq) + "\n")

    o.close()

    with open("NEFrequencies.txt", "w") as f:

        for (word, freq) in ne_freqs:
            word = word.encode("utf-8")
            f.write(str(word) + " " + str(freq) + "\n")

    f.close()
```

```
def compare_texts(freqs, name):


    text_anonym = open(name+"Anonym.txt", "r")

    text_anonym = text_anonym.read()

    text_anonym = text_anonym.decode("utf-8")

    text_anonym = nltk.word_tokenize(text_anonym)

    false_positives_list = []

    false_negatives_list = []

    total_errors_list = []

    accuracy_list = []

    precision_list = []

    recall_list = []

    f1_list = []

    f2_list = []

    total_positives_list = []

    total_negatives_list = []

    for j in range(len(freqs)):

        freq = freqs[j]

        text_freq = open(str(name)+"Version1Freq" + str(freq) +
".txt", "r")

        text_freq = text_freq.read()

        text_freq = text_freq.decode("utf-8")

        text_freq = nltk.word_tokenize(text_freq)

        true_positives_text = len([i for i in
range(len(text_anonym)) if text_freq[i] == "***" and text_anonym[i]
== "***"])
        true_negatives_text = len([i for i in
range(len(text_anonym)) if text_anonym[i] != "***" and text_freq[i]
!= "***"])
        false_positives_text = len([i for i in
range(len(text_anonym)) if text_anonym[i] != "***" and text_freq[i]
=="***"])
        false_negatives_text = len([i for i in
range(len(text_anonym)) if text_anonym[i] == "***" and text_freq[i]
!= "***"])
```

```
        accuracy = (true_positives_text + true_negatives_text) /
(true_positives_text + true_negatives_text + false_positives_text +
false_negatives_text)

        accuracy = accuracy*100

        precision = (true_positives_text) / (true_positives_text +
false_positives_text)

        precision = precision * 100

        recall = true_positives_text / (true_positives_text +
false_negatives_text)

        recall = recall * 100

        f1_score = 2*(recall*precision)/(recall + precision)

        f2_score = 5 * recall * precision / (4*precision+recall)

        false_positives = (false_positives_text /
len(text_anonym))*100

        false_negatives = (false_negatives_text / len(text_anonym))
* 100

        total_errors = ((false_positives_text +
false_negatives_text) / len(text_anonym))*100

        false_positives_list.append(false_positives)

        false_negatives_list.append(false_negatives)

        total_errors_list.append(total_errors)

        accuracy_list.append(accuracy)

        precision_list.append(precision)

        recall_list.append(recall)

        f1_list.append(f1_score)

        f2_list.append(f2_score)

        total_positives_list = (true_positives_text +
false_negatives_text)/len(text_anonym)

        total_negatives_list = (true_negatives_text +
false_positives_text)/len(text_anonym)

    return total_positives_list, total_negatives_list,
false_positives_list, false_negatives_list, total_errors_list,
accuracy_list, precision_list, recall_list, f1_list, f2_list
```

```
def test_tool_2(text, threshold_freq_ne, threshold_freq_np):

    name = "EloiseGerry"

    text_words = nltk.word_tokenize(text)

    persons, organizations, gpes, facilities =
get_classified_named_entities(text)

    ne = organizations + gpes + facilities

    ne = list(set(ne)- set(persons))

    list_of_ne = list(set(ne))

    list_of_np = text_to_np_list(text)

    list_of_np = list(set(list_of_np))

    with open("NPFrequencies.txt", "r") as f:

        np_freqs = []

        for line in f.readlines():
            element = line.split()
            word = element[0]
            freq = element[1]
            word = word.decode("utf-8")
            freq = float(freq)
            np_freqs.append([word,freq])

    f.close()

    with open("NEFrequencies.txt", "r") as f:

        ne_freqs = []

        for line in f.readlines():
            element = line.split()
            word = element[0]
            freq = element[1]
            word = word.decode("utf-8")
            freq = float(freq)
            ne_freqs.append([word,freq])

    f.close()

    old_ne = [w for (w,freq) in ne_freqs]

    old_np = [w for (w,freq) in np_freqs]

    new_np = list(set(list_of_np) - set(old_np))

    new_ne = list(set(list_of_ne) - set(old_ne))

    new_ne_freqs = get_freqs(new_ne)
```

```
    new_np_freqs = get_freqs(new_np)

    ne_freqs = ne_freqs + new_ne_freqs

    np_freqs = np_freqs + new_np_freqs

    ne_to_anonym = get_words_to_anonymize(ne_freqs,
threshold_freq_ne)

    np_to_anonym = get_words_to_anonymize (np_freqs,
threshold_freq_np)

    words_to_remove_list = persons + ne_to_anonym + np_to_anonym

    words_to_remove_list = list(set(words_to_remove_list))

    words_anonym = ["***" if w in words_to_remove_list
                else w
                for w in text_words]

    text = wordlist_to_writtable_text(text_words)

    text_anonym = wordlist_to_writtable_text(words_anonym)

    f = open(name+".txt","w")

    f.write(text)

    f.close()

    g = open(name+"Version2Freqs" + str(threshold_freq_ne) + "_" +
str(threshold_freq_np)+ ".txt", "w")

    g.write(text_anonym)

    g.close()

    with open("NPFrequencies.txt", "w") as o:

        for (word, freq) in np_freqs:
            word = word.encode("utf-8")
            o.write(str(word) + " " + str(freq) + "\n")

    o.close()

    with open("NEFrequencies.txt", "w") as f:

        for (word, freq) in ne_freqs:
            word = word.encode("utf-8")
            f.write(str(word) + " " + str(freq) + "\n")

    f.close()


def compare_texts_2freqs (freqs_ne, freqs_np):

    name = "EloiseGerry"
```

```
    text_anonym = open(name+"Anonym.txt", "r")
    text_anonym = text_anonym.read()
    text_anonym = text_anonym.decode("utf-8")
    text_anonym = nltk.word_tokenize(text_anonym)
    false_positives_matrix = []
    false_negatives_matrix =  []
    total_errors_matrix =  []
    accuracy_matrix = []
    precision_matrix = []
    recall_matrix = []
    f1_matrix = []
    f2_matrix = []
    for k in range(len(freqs_np)):

        freq_np = freqs_np[k]
        false_positives_list = []
        false_negatives_list =  []
        total_errors_list =  []
        accuracy_list = []
        precision_list = []
        recall_list = []
        f1_list = []
        f2_list = []
        for j in range(len(freqs_ne)):
            freq_ne = freqs_ne[j]
            text_freq = open(name+"Version2Freqs" + str(freq_ne) +
"_" + str(freq_np)+ ".txt", "r")
            text_freq = text_freq.read()
            text_freq = text_freq.decode("utf-8")
            text_freq = nltk.word_tokenize(text_freq)
```

```
            true_positives_text = len([i for i in
range(len(text_anonym)) if text_freq[i] == "***" and text_anonym[i]
== "***"])
            true_negatives_text = len([i for i in
range(len(text_anonym)) if text_anonym[i] != "***" and text_freq[i]
!= "***"])
            false_positives_text = len([i for i in
range(len(text_anonym)) if text_anonym[i] != "***" and text_freq[i]
=="***"])
            false_negatives_text = len([i for i in
range(len(text_anonym)) if text_anonym[i] == "***" and text_freq[i]
!= "***"])

            accuracy = (true_positives_text + true_negatives_text)
/ (true_positives_text + true_negatives_text + false_positives_text
+ false_negatives_text)

            accuracy = accuracy*100

            precision = (true_positives_text) /
(true_positives_text + false_positives_text)

            precision = precision * 100

            recall = true_positives_text / (true_positives_text +
false_negatives_text)

            recall = recall * 100

            f1_score = 2*(recall*precision)/(recall + precision)

            f2_score = 5 * (recall * precision) /
(4*precision+recall)

            false_positives = (false_positives_text /
len(text_anonym))*100

            false_negatives = (false_negatives_text /
len(text_anonym)) * 100

            total_errors = ((false_positives_text +
false_negatives_text) / len(text_anonym))*100

            false_positives_list.append(false_positives)

            false_negatives_list.append(false_negatives)

            total_errors_list.append(total_errors)

            accuracy_list.append(accuracy)

            precision_list.append(precision)

            recall_list.append(recall)

            f1_list.append(f1_score)

            f2_list.append(f2_score)
```

```
        false_positives_matrix.append(false_positives_list)

        false_negatives_matrix.append(false_negatives_list)

        total_errors_matrix.append(total_errors_list)

        accuracy_matrix.append(accuracy_list)

        precision_matrix.append(precision_list)

        recall_matrix.append(recall_list)

        f1_matrix.append(f1_list)

        f2_matrix.append(f2_list)


    return false_positives_matrix, false_negatives_matrix,
total_errors_matrix, accuracy_matrix, precision_matrix,
recall_matrix, f1_matrix, f2_matrix
```

final_gui.py

```
from Tkinter import *
import Tkinter, tkFileDialog
import urllib2
import nltk
from bs4 import BeautifulSoup
import re
from functionsfile import get_text_from_html , text_to_np_list ,
get_freq, generalize_hyper, get_classified_named_entities ,
get_freqs , get_words_to_anonymize , choose_words_to_anonymize
,wordlist_to_writtable_text
from nltk.corpus import wordnet as wn


font10 = "-family Arial -size 18 -weight normal -slant roman "  \
            "-underline 0 -overstrike 0"
font11 = "-family Arial -size 9 -weight normal -slant roman "  \
            "-underline 0 -overstrike 0"

fontquestion = ("Arial", 9, "bold")

root1 = Tk()
root1.title("Anonymization tool: Welcome window")

def openfile():
    filename = tkFileDialog.askopenfilename(initialdir =
"C:\Users\Helia\Documents\TFG\Biographies" ,filetypes = [("text
file", "*.txt")])
    global text
    f = open(filename)
    f = f.read()
    text = f.decode("utf-8")
    root1.destroy()
    return text
```

```
frame = Frame(root1, background="#97c9d8", relief='groove',
borderwidth="2", width=500, height=300).pack()

Button(frame, text = "Open file", command =
openfile).place(relx=0.437, rely=0.59, height=24, width=61)

message1 = Message(frame,background="#97c9d8",font =
font10,highlightbackground="#97c9d8",foreground="#000000",
text='Welcome to the anonymization tool', width=410,
justify='center').place(relx=0.114, rely=0.131, relheight=0.239
                , relwidth=0.779)

message2 = Message(frame,background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000",text =
'''Please select the text file you would like to anonymize.''',
width = 250, justify='center').place(relx=0.247, rely=0.426,
relheight=0.108
                , relwidth=0.475)
root1.mainloop()


text_words = nltk.word_tokenize(text)

persons, organizations, gpes, facilities =
get_classified_named_entities(text)

ne = organizations + gpes + facilities

list_of_ne = list(set(ne))

list_of_np = text_to_np_list(text)

list_of_np = list(set(list_of_np))

with open("NPFrequencies.txt", "r") as f:

    np_freqs = []

    for line in f.readlines():
        element = line.split()
        word = element[0]
        freq = element[1]
        word = word.decode("utf-8")
        freq = float(freq)
        np_freqs.append([word,freq])

f.close()

with open("NEFrequencies.txt", "r") as f:

    ne_freqs = []

    for line in f.readlines():
        element = line.split()
        word = element[0]
        freq = element[1]
        word = word.decode("utf-8")
```

```
        freq = float(freq)
        ne_freqs.append([word,freq])

f.close()

old_ne = [w for (w,freq) in ne_freqs]

old_np = [w for (w,freq) in np_freqs]

new_np = list(set(list_of_np) - set(old_np))

new_ne = list(set(list_of_ne) - set(old_ne))

new_ne_freqs = get_freqs(new_ne)

new_np_freqs = get_freqs(new_np)

ne_freqs = ne_freqs + new_ne_freqs

np_freqs = np_freqs + new_np_freqs


root2 = Tk()
root2.title("Anonymization tool: privacy level")

var = IntVar()
var.set(1)

def quit_loop2():
    global selection
    selection = var.get()
    root2.destroy()

frame = Frame(root2, background="#97c9d8",
relief='groove',borderwidth="2", width=250,
height=200).grid(rowspan = 10, columnspan = 1)
Label(frame, text = "Select the desired privacy level", background
= "#97c9d8",font =
fontquestion,highlightbackground="#97c9d8",foreground="#000000",
justify = "center").grid(row=0)
Label(frame,text = "1 for the lowest, 5 for the highest",
background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000", justify
= "center").grid(row=1)
Radiobutton(frame, text = "Level 1", variable=var, value = 1,
background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=3)
Radiobutton(frame, text = "Level 2", variable=var, value = 2,
background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=4)
Radiobutton(frame, text = "Level 3", variable=var, value = 3,
background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=5)
Radiobutton(frame, text = "Level 4", variable=var, value = 4,
background = "#97c9d8",font =
```

```
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=6)
Radiobutton(frame, text = "Level 5", variable=var, value = 5,
background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=7)
Button(frame, text = "Select", command=quit_loop2).grid(row=8)
root2.mainloop()

if selection == 5:
    threshold_freq_ne = 200
    threshold_freq_np  = 5
elif selection == 4:
    threshold_freq_ne = 175
    threshold_freq_np = 4
elif selection == 3:
    threshold_freq_ne = 150
    threshold_freq_np = 3
elif selection == 2:
    threshold_freq_ne = 125
    threshold_freq_np = 2
elif selection == 1:
    threshold_freq_ne = 100
    threshold_freq_np = 1

ne_to_anonym = get_words_to_anonymize(ne_freqs, threshold_freq_ne)

np_to_anonym = get_words_to_anonymize (np_freqs, threshold_freq_np)

np_to_anonym = list(set(np_to_anonym) - set(ne_to_anonym))

def quit_loop3():
    global yesorno
    yesorno = varyesno.get()
    root3.destroy()

def quit_loop4():
    global anonymchoice
    anonymchoice = varanonymchoice.get()
    root4.destroy()

def quit_loop5():
    global meaning
    meaning = varmeaning.get()
    root5.destroy()

def quit_loop6():
    global hyper
    hyper = varhyper.get()
    root6.destroy()

def quit_loop7():
    global lemma
    lemma = varlemma.get()
    root7.destroy()

def quit_loop8():
    global hypermean
```

```
    hypermean = varhypermean.get()
    root8.destroy()

def quit_loop9():
    global hyperhyper
    hyperhyper = varhyperhyper.get()
    root9.destroy()

def quit_loop10():
    global hyperlemma
    hyperlemma = varhyperlemma.get()
    root10.destroy()

np_hyper = []

word_sent =[]

sents = []

word_indexes = []

text_sents = nltk.sent_tokenize(text)
text_words1 = [nltk.word_tokenize(sent) for sent in text_sents]

for word in np_to_anonym:
    for i in range(len(text_words1)):
        indices = [j for j, w in enumerate(text_words1[i]) if w ==
word]
        word_sent = word_sent + ([[word, text_sents[i]]] *
len(indices))

for (word, sent) in word_sent:
    root3 = Tk()
    root3.title("Anonymization tool")
    root3.configure(background = "#97c9d8")
    varyesno = IntVar()
    varyesno.set(1)

    frame = Frame(root3, background="#97c9d8",
relief='groove',borderwidth="2").grid(rowspan = 10, columnspan = 1)
    Label(frame, text = "Word: " +str(word.encode("utf-
8")),background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=2)
    Label(frame, text = "Sentence: " + str(sent.encode("utf-
8")),background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=3)
    Label(frame, text = "Do you want to obfuscate this
word?",background = "#97c9d8",font =
fontquestion,highlightbackground="#97c9d8",foreground="#000000").gr
id(row=1)
    Radiobutton(frame, text = "Yes", variable = varyesno, value =
1,background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=4)
    Radiobutton(frame, text = "No", variable = varyesno, value = 2,
background = "#97c9d8",font =
```

```
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=5)
    Button(frame, text = "Select", command=quit_loop3, font =
font11).grid(row=6)
    root3.mainloop()

    if yesorno == 1:
        root4 = Tk()
        root4.title("Anonymization tool")
        root4.configure(background = "#97c9d8")
        varanonymchoice = IntVar()
        varanonymchoice.set(1)
        Label(root4, text = "(Word: " +str(word.encode("utf-
8"))+")",background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=2)
        Label(root4, text = "Plese choose an obfuscation method for
this word.",background = "#97c9d8",font =
fontquestion,highlightbackground="#97c9d8",foreground="#000000").gr
id(row=1)
        Radiobutton(root4, text = "Black it out", variable =
varanonymchoice, value = 1, background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=4)
        Radiobutton(root4, text = "Generalize it (if possible)",
variable = varanonymchoice, value = 2,background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=5)
        Button(root4, text = "Select",
command=quit_loop4).grid(row=6)
        root4.mainloop()

        if anonymchoice == 1:
            general = "***"
            np_hyper.append([word, general])
        elif anonymchoice == 2:
            synsets = wn.synsets(str(word))
            if synsets ==[]:
                general = "***"
                np_hyper.append([word,general])

            else:
                root5 = Tk()
                root5.title("Anonymization tool")
                root5.configure(background = "#97c9d8")
                varmeaning = IntVar()
                varmeaning.set(1)
                Label(root5,text =  "These are the different
meanings the word may have. Please select the adequate.",
background = "#97c9d8",font =
fontquestion,highlightbackground="#97c9d8",foreground="#000000").gr
id(row=0)
                Label(root5, text = "(Word: " +
str(word.encode("utf-8"))+")",background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=1)
                for i in range(len(synsets)):
```

```
                    Radiobutton(root5, text = str(synsets[i])+ ": "
+ str(synsets[i].definition().encode("utf-8")), var = varmeaning,
value = i,background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=2+i, sticky = W)
                Radiobutton(root5, text = "None of the others:
Black it out", variable = varmeaning, value = 100,background =
"#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
= 2+len(synsets), sticky = W)
                Button(root5, text = "Select", command =
quit_loop5).grid(row = 2+len(synsets)+1 )
                root5.mainloop()
                if meaning == 100:
                    general = "***"
                    np_hyper.append([word, general])
                else:
                    w = synsets[i]
                    hyps = w.hypernyms()
                    if hyps == []:
                        general = "***"
                        np_hyper.append([word,general])
                    else:
                        if len(hyps) > 1:
                            root6 = Tk()
                            root6.title("Anonymization tool")
                            root6.configure(background = "#97c9d8")
                            varhyper = IntVar()
                            varhyper.set(1)
                            Label(root6, text = "These are the
different hypernyms the word may have.Please select the
adequate.",background = "#97c9d8",font =
fontquestion,highlightbackground="#97c9d8",foreground="#000000").gr
id(row=0)
                            Label(root6, text = "(Word: " +
str(word.encode("utf-8"))+")",background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=1)
                            for i in range(len(hyps)):
                                Radiobutton(root6, text =
str(hyps[i]), variable = varhyper, value = i, background =
"#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
= 2+i, rowspan = len(hyps))
                            Button(root6, text = "Select", command
= quit_loop6).grid(row = 3+len(hyps))
                            root6.mainloop()
                            lemma_hyp = hyps[hyper].lemmas()
                        else:
                            lemma_hyp = hyps[0].lemmas()


                        if len(lemma_hyp) > 1:
                            root7 = Tk()
                            root7.title("Anonymization tool")
                            root7.configure(background = "#97c9d8")
                            varlemma = IntVar()
                            varlemma.set(1)
```

```
                            Label(root7, text = "These are the
different lemmas the hypernym may have. Please select the
adequate.",background = "#97c9d8",font =
fontquestion,highlightbackground="#97c9d8",foreground="#000000").gr
id(row=0)
                            Label(root7, text = "(Word: " +
str(word.encode("utf-8"))+")",background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=1)
                            for i in range(len(lemma_hyp)):
                                Radiobutton(root7, text =
str(lemma_hyp[i].name()), variable = varlemma, value = i,
background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
= 2+i)
                            Button(root7, text = "Select", command
= quit_loop7).grid(row = 2+len(lemma_hyp))
                            root7.mainloop()
                            general = lemma_hyp[lemma].name()
                            general = re.sub("_", " ", general)
                            general = general.upper()
                            np_hyper.append([word, general])

                        else:
                            general = lemma_hyp[0].name()
                            general = re.sub("_"," ", general)
                            general = general.upper()
                            np_hyper.append([word,general])



    elif yesorno == 2:
        general = word
        np_hyper.append([word,general])


for j in range(len(np_hyper)):
    hyper = np_hyper[j][1]
    if hyper.isupper():
        freq = get_freq(hyper)
        general = hyper
        while freq < threshold_freq_np:
            general = re.sub(" ", "_", general)
            synsets = wn.synsets(str(general))
            if len(synsets) > 1:
                root8 = Tk()
                root8.title("Anonymization tool")
                root8.configure(background = "#97c9d8")
                varhypermean = IntVar()
                varhypermean.set(1)
                Label(root8, text = "It looks like some hypernyms
need more generalization.", background = "#97c9d8",font =
fontquestion,highlightbackground="#97c9d8",foreground="#000000").gr
id(row=0)
                Label(root8, text = "Which synset does this word
refer to? Please select the adequate.", background = "#97c9d8",font
=
```

```
fontquestion,highlightbackground="#97c9d8",foreground="#000000").gr
id(row=2)
                Label(root8, text = "(Word (hypernym): " +
str(word.encode("utf-8"))+")",background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=3)
                for i in range(len(synsets)):
                    Radiobutton(root8, text = str(synsets[i])+ ": "
+ str(synsets[i].definition().encode("utf-8")), var = varhypermean,
value = i, background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=3+i, sticky = W)
                Button(root8, text = "Select", command =
quit_loop8).grid(row = 4+len(synsets))
                root8.mainloop()
                w = synsets[hypermean]
            else:
                w = synsets[0]
            hyps = w.hypernyms()
            if len(hyps) > 1:
                root9 = Tk()
                root9.title("Anonymization tool")
                root9.configure(background = "#97c9d8")
                varhyperhyper = IntVar()
                varhyperhyper.set(1)
                Label(root9, text = "Which hypernym does your word
refer to? Please select the adequate", background = "#97c9d8",font
=
fontquestion,highlightbackground="#97c9d8",foreground="#000000").gr
id(row=0)
                Label(root9, text = "(Word (hypernym): " +
str(word.encode("utf-8"))+")",background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=1)
                for i in range(len(hyps)):
                    Radiobutton(root9, text = str(hyps[i]),
variable = varhyperhyper, value = i, background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=2+i)
                Button(root9, text = "Select", command =
quit_loop9).grid(row = 2+len(hyps))
                root9.mainloop()
                lemma_hyp = hyps[hyperhyper].lemmas()
            else:
                lemma_hyp = hyps[0].lemmas()
            if len(lemma_hyp) > 1:
                root10 = Tk()
                root10.title("Anonymization tool")
                root10.configure(background = "#97c9d8")
                varhyperlemma = IntVar()
                varhyperlemma.set(1)
                Label(root10, text = "Which lemma hypernym does
your word refer to? Please select the adequate.",background =
"#97c9d8",font =
fontquestion,highlightbackground="#97c9d8",foreground="#000000").gr
id(row = 0)
                Label(root9, text = "(Word (hypernym): " +
str(word.encode("utf-8"))+")",background = "#97c9d8",font =
```

```
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=1)
                for i in range(len(lemma_hyp)):
                    Radiobutton(root10, text =
str(lemma_hyp[i].name()), variable = varhyperlemma, value =
i,background = "#97c9d8",font =
font11,highlightbackground="#97c9d8",foreground="#000000").grid(row
=2+i)
                Button(root10,text = "Select", command =
quit_loop10).grid(row = 2+len(lemma_hyp))
                root10.mainloop()
                general = lemma_hyp[hyperlemma].name()
                general = re.sub("_", " ", general)
                freq = get_freq(general)
                general = general.upper()
                np_hyper[j][1] = general
            else:

                general = lemma_hyp[0].name()
                general = re.sub("_", " ", general)
                freq = get_freq(general)
                general = general.upper()
                np_hyper[j][1] = general


words_to_remove_list =  persons + ne_to_anonym

words_to_remove_list = list(set(words_to_remove_list))


words_anonym = ["***" if w in words_to_remove_list
                else w
                for w in text_words]


word_text = []
word_list = []

for (word, gen) in np_hyper:
    indices_in_text = [i for i, w in enumerate(words_anonym) if w
== word]
    indices_in_list = [i for i, w in enumerate(np_hyper) if w[0] ==
word]
    word_text.append([word, indices_in_text])
    word_list.append([word, indices_in_list])
for j in range(len(word_text)):
    item1 = word_text[j]
    item2 = word_list[j]
    for i in range(len(item1[1])):
        index_text = item1[1][i]
        index_list = item2[1][i]
        words_anonym[index_text] = np_hyper[index_list][1]

#text = wordlist_to_writtable_text(text_words)

text_anonym = wordlist_to_writtable_text(words_anonym)

g = open("TextprovaAnonym.txt", "w")
```

```
g.write(text_anonym)

g.close()

with open("NPFrequencies.txt", "w") as o:

    for (word, freq) in np_freqs:
        word = word.encode("utf-8")
        o.write(str(word) + " " + str(freq) + "\n")

o.close()


with open("NEFrequencies.txt", "w") as f:

    for (word, freq) in ne_freqs:
        word = word.encode("utf-8")
        f.write(str(word) + " " + str(freq) + "\n")

f.close()
```