UPPSALA
UNIVERSITET

UPTEC F 15054
Examensarbete 30 hp
September 2015

Automatic de-identification of case narratives from spontaneous reports in VigiBase

Jakob Sahlström

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Automatic de-identification of case narratives from spontaneous reports in VigiBase

Jakob Sahlström

The use of patient data is essential in research but it is on the other hand confidential and can only be used after acquiring approval from an Ethical Board and informed consent from the individual patient. A large amount of patient data is therefore difficult to obtain if sensitive information, such as names, id numbers and contact details, are not removed from the data, by so called de-identification. Uppsala Monitoring Centre maintains the world's larges database of individual case reports of any suspected adverse drug reaction. There exists, of today, no method for efficiently de-identifying the narrative text included in these which causes countries like the United States of America reports to exclude the narratives in the reports.

The aim of this thesis is to develop and evaluate a method for automatic de-identification of case narratives in reports from the WHO Global Individual Case Safety Report Database System, VigiBase. This report compares three different models, namely Regular Expressions, used for text pattern matching, and the statistical models Support Vector Machine (SVM) and Conditional Random Fields (CRF). Performance, advantages and disadvantages are discussed as well as how identified sensitive information is handled to maintain readability of the narrative text. The models developed in this thesis are also compared to existing solutions to the de-identification problem.

The 400 reports extracted from VigiBase were already well de-identified in terms of names, ID numbers and contact details, making it difficult to train statistical models on these categories. The reports did however, contain plenty of dates and ages. For these categories Regular Expression would be sufficient to achieve a good performance. To identify entities in other categories more advanced methods such as the SVM and CRF are needed and will require more data. This was prominent when applying the models on the more information rich i2b2 de-identification challenge benchmark data set where the statistical models developed in this thesis performed at a competing level with existing models in the literature.

Handledare: Johan Ellenius
Ämnesgranskare: Sofia Cassel
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F 15054

Contents

1 Introduction

Using patient data in research is crucial to get new relevant insights into how humans are affected by their environment and to develop methods for prevention of diseases and disorders. Diversity and complexity of the human body makes it impossible to generate synthetic data that is representative to some population in general. Patient data, however, can be difficult to collect since typically, an Ethical Board has to approve the use of the medical record as well as obtain informed consent from the individual patient. However, if the data is de-identified then these requirements do not necessarily apply.

Uppsala Monitoring Centre (UMC) is an independent foundation with the primary goal of improving patient safety and the safety and effectiveness of medicine usage in all corners of the world. UMC maintains and analyzes the world's largest data base of individual case reports of any suspected unintentional effects from drugs, so called *Adverse Drug Reaction* (ADR). The data base is named VigiBase [1]. These reports contain both structured non-sensitive information such as patient's year of birth and suspected ADRs, as well as free text narratives that may contain sensitive information that can identify an individual patient. Reports of suspected ADRs from countries like the United States of America do not include the narrative text since UMC cannot, at the moment, guarantee the narratives being de-identified. A method for automatic de-identification before storing the free text in VigiBase would make it possible for countries like the U.S. to send complete reports and not exclude the narratives. Reports coming from the U.S. cover about 50% of all spontaneous reports retrieved by UMC. Obtaining the narrative information from these reports containing limited information would provide valuable data for the research and signal detection team at UMC to improve the safety of drug usage. Narrative information in addition to the structured fields is of great importance to not make incorrect interpretations of the reports which could result in wrong regulatory decisions [2].

According to the United States Health Insurance Portability and Accountability Act (HIPAA) [3] there are 18 data elements, called Protected Health Information (PHI), that have to be removed from a clinical record for it to be considered de-identified, see Figure 1.

One essential variable in the causality assessment of a suspected ADR is the time interval between medical treatments and the onset of an adverse event. In the narratives this information is often stated as dates when a medical treatment was initialized. Specific dates are, on the other hand, sensitive information according to HIPAA and must therefore be removed in the de-identification process. This issue can be solved by replacing the dates with time intervals from a reference point or add a document-specific random offset to all dates. In this way the specific dates are removed and the time intervals are preserved. The same methodology can be applied to other sensitive elements such as names and locations. The effect of de-identifying a text is illustrated in the example below.

1. Names
2. All geographic subdivisions smaller than a State, including street address, city, county, precinct, zip code or equivalents except for the initial three digits of a zip code if the corresponding area contains more than 20.000 people.
3. All elements of dates (except year) for dates directly related to an individual, including birth date, admission date, discharge date, date of death and all ages over 89 and all elements of dates indicative of such age
4. Telephone numbers
5. Fax numbers
6. Electronic mail addresses
7. Social security numbers
8. Medical record numbers
9. Health plan beneficiary numbers
10. Account numbers
11. Certificate/license numbers
12. Vehicle identifiers and serial numbers, including license plate numbers
13. Device identifiers and serial numbers
14. Web Universal Resource Locators (URLs)
15. Internet Protocol (IP) address numbers
16. Biometric identifiers, including finger and voice prints
17. Full face photographic images and any comparable images
18. Any other unique identifying number, characteristic, or code

Figure 1: Protected Health Information (PHI) to be removed from a text to be classified as de-identified (as defined by United States Health Insurance Portability and Accountability Act (HIPAA)).

Original Text:

> Mr. Smith visited Uppsala Hospital at May 1 2014. Mr. Smith later experienced symptoms on May 16 2014.

De-identified text:

> [PERSON] visited [LOCATION] at [14 September 2012]. [PERSON] later experienced symptoms on [29 September 2014].

The objective of this study is to compare models with different complexity and evaluate their performance on de-identifying narrative information in VigiBase reports. The process of de-identification can be divided into two parts:

1. Identify sensitive information
2. Reduce information loss by replacing identified entities with informative substitutes.

On one hand, when de-identifying a report it is crucial that all sensitive entities are found. A missed date, for example, could help re-identifying other masked dates in the text. On the other hand, to reduce the loss of information of a de-identified report, high precision and informative substitutions are needed to ensures that no unnecessary text is removed and that the text maintains its readability even though sensitive elements are removed.

Words, numbers and punctuations, in this report collectively named as *tokens*, can have different meaning depending on the context they occur in and the same goes for sensitivity of a sequence of tokens. A disease, such as Parkinson's disease, could in specific

contexts be interpreted as a name of a person. To get a computer to distinguish between sensitive and non-sensitive entities in a text is not a trivial task and requires detailed analysis to achieve good performance.

2 Related Work

A common task in Natural Language Processing (NLP) is to predict the Part-of-Speech (POS) tag [4] for each token in a sentence. Publicly available corpora containing a large amount of text that have been manually annotated with POS tags are often used to train these taggers. One of the biggest corpora is the Penn Treebank [5] corpus which consists of over 4.5 million manually annotated words from the American English language and is often used as a benchmark. State-of-the-art POS taggers typically have an accuracy of around 97% when trained and evaluated on the Penn Treebank *Wall Street Journal* corpus [6].

For a statistical model to be able to find patterns in the data we need to describe the data in a way that is suitable for the model. This is done by defining so called *features* which in some way describes a data point. An example of what kind of features are often used in the area of NLP is presented by Toutanova et al. [7, 8] at Stanford, authors of a widely used POS tagger. Their POS tagger uses features generated directly from the token itself, such as the current token, next token, suffixes and prefixes as well as boolean features telling if the current token contains a number, hyphen or uppercase characters. These features can be seen as *local* since they concern the target token and its immediate surroundings. Other features may include lookups in external resources, such as dictionaries, and features based on characteristics of the bigger context in which the target token exist, also referred to as *non-local* features, e.g. number of tokens in the sentence and position in document [9].

Another task in NLP is Named-Entity Recognition (NER) where the goal is to label elements of a text with pre-defined categories, e.g. names of persons, location or organizations. NER can be seen as two tasks: detecting entities, and classifying the entities detected. Entity detection is often referred to as *chunking* and is usually solved by combining tokens to phrases using a model based on the tokens and their POS tags. [10, 11]

There are mainly two approaches to a NER problem. One is to include linguistic grammar-based techniques, requiring extensive manual work from linguists. The other is to use statistical models which usually reduce the need of linguistic knowledge but require a vast amount of manually annotated data [12]. A statistical NER model often uses the POS tag, from applying a pre-trained POS tagger, as a feature. Using POS tags as features result in one indication function per POS tag specifying if the token represents a verb, noun, etc. In addition to the POS tag, features like the ones described in [7, 8] are also included. The process of de-identifying medical records can be seen as a NER task since the goal is to identify elements of the text that belong to the Protected Health Information (PHI) categories defined by HIPAA.

The words *anonymization* and *de-identification* are often used as equivalents, though there exists an important distinction between the two terms. Clete A. Kushida et al. [13] state that de-identification of medical records is the act of removing or replacing

personal identifiers, making it difficult to restore the connection between the individual and his or her data. However, de-identified data sets are allowed to contain encrypted identifiers where only authorized individuals have access to the encryption key. The existence of a key makes it possible to reestablish a link between individual and data for an individual with correct authorization. The data set must not contain any data that would allow unauthorized individuals to reestablish this link. Anonymization on the other hand, is referred to as irreversibly removing all links between data and individual to the extent that it is virtually impossible to restore the connection between the individual and his or her medical record.

Meystre et al. [14] review systems for automatic de-identification of narrative text in electronic health records. The paper brings up 18 different methods used in the area of text de-identification, some mainly based on pattern matching and/or rule-based techniques and some mainly based on machine learning techniques. For each method, the authors make a detailed analysis in terms of the architecture used, the PHI categories detected, what external sources were used and the type of clinical documents targeted by the method. Meystre et al. found that the majority of these methods relied only on pattern matching, rules and dictionaries.

Informatics for Integrating Biology and the Bedside (i2b2) [15] is a center for Biomedical Computing based at Partners HealthCare System. They have repeatedly provided NLP challenges in the area of clinical research where participants received fully de-identified documents and an objective. The i2b2 challenge in 2006 focused on de-identification [16] and is now frequently used as a benchmark when comparing models for the de-identification task. The i2b2 data set for the de-identification challenge will be discussed in detail in Section 5.2.2. Other challenges include extracting medications, identifying obesity and identifying risk factors for heart diseases.

Aramaki et al. participated in the i2b2 de-identification challenge and developed a system which not only uses local features (e.g. target and surrounding tokens) but also takes external sources into account (such as dictionaries) as well as non-local features (e.g. sentence length and position within the document) [9]. The use of non-local features was based on the insights that sentences including PHI occurred at the beginning or end of a document and were in general shorter in length compared to sentences not containing PHI elements. The system was based on a statistical machine learning technique called Conditional Random Fields (CRF) which takes the context of a token into account.

Ferrández et al. [17] present a solution to the task of de-identifying medical records by using a two step process after first pre-processing the data.

1. High sensitivity extraction including dictionary lookups, pattern matching and the prediction from a CRF model to determine the most probable PHI category for a token.

2. PHI candidates identified in the previous step were used to train individual binary classifiers for each PHI category using a machine learning technique called Support Vector Machines (SVM). The target variable was to predict if the resulting annotations from previous step were correct or not.

In this way Ferrández et al. managed to reduce the number of elements incorrectly classified as sensitive while keeping a high confidence in the classification.

The time between drug exposure and the occurrence of a medical event is one of the most important aspects when determining the likelihood of the event being caused by the drug or not. However, certain dates are, according to HIPAA, categorized as PHI and should be removed. Thus it is important to retain as much information as possible. One solution to not lose date information in de-identified reports is applied by the De-ID system developed by University of Pitsburgh and evaluated by D. Gupta [18]. De-ID offers the functionality of adding the same random date offset to all dates in a report. In this way the actual dates are modified and the interval and granularity of a date are retained.

3 Aim

The main objective of this report is to develop and evaluate a method for automatic de-identification of case narratives in reports from the WHO Global Individual Case Safety Report Database System, VigiBase. The de-identified narratives should maintain readability by replacing as much sensitive information as possible with informative substitutes. This report compares three different methods with varying complexity to find out which approach is the most suitable to fulfill this objective. The methods, listed in order of increasing complexity, are Regular Expressions, Support Vector Machines and Conditional Random Fields. Furthermore, we want to obtain an answer to the question *How does our model compare to existing models?*

Grammar and sentence structure differs a lot between languages, so to develop a multilingual algorithm requires deep knowledge in linguistics in addition to scientific computing and machine learning. Combining this with UMC's goal of retrieving the missing narratives of the U.S. reports, this thesis is limited to English narratives only.

4 Background

In this chapter we explain the main principles and methods that are used in this thesis. We will bring up the basic Regular Expressions used for text pattern matching, the statistical models Support Vector Machines and Conditional Random Fields, as well as how the statistical models are trained to extract entities of interest from a text.

4.1 Regular Expressions

A Regular Expression (RegEx) [19] is a combination of characters describing a span of text that follows a certain pattern. RegEx are used in many of the search and replace functions encountered in most popular text editors. A simple example is the task of finding all entries of a word that can have different spellings, such as the Swedish surname *Jönsson*, which could be spelled as *Jönsson, Jonsson* or *Joensson* depending on the situation. Instead of searching a text for each variant separately the RegEx `J(ö|oe?)nsson` will match all three spellings. This example covers three of the basic concepts of regular expressions:

OR operator The vertical bar separates alternatives, e.g. `gray|grey` matches `gray` and `grey`.

Grouping Parentheses are used to define a scope for the operators within the parentheses, e.g. `gr(a|e)y` is equivalent to `gray|grey`

Quantification Quantifiers specify how many repetitions a pattern is allowed. A quantifier can be applied to a single character or a group. The following are the most common quantifiers:

- `?` : Question mark indicates that the preceding element occurs *zero or one* time, e.g. `honou?r` matches both `honor` and `honour`.
- `*` : Asterix indicates that the preceding element occurs *zero or more* times, e.g. `39*5` matches `35`, `395`, `3995` etc.
- `+` : Plus sign indicates that the preceding element occurs *one or more* times, e.g. `39+5` matches `395`, `3995`, `39995` etc. Note that `35` is *not* matched.
- {,} : It is possible to set at specific number of repetitions by specifying an interval inside curly brackets, e.g. `39{2,4}5` matches `3995`, `39995` and `399995` only.

Predefined character groups are often used to simplify a regular expression and make it more readable. Appendix A lists common groups used in regular expressions. Using the basic concepts of RegEx explained here, patterns can be used to identify entities that are known to follow a certain pattern, such as dates. In Section 5.5 we explain what a RegEx for dates and ages can look like.

4.2 Support Vector Machines

Classification is the process of assigning a category, or class, to a new observation from a predefined set of classes. When talking about binary classification the set of classes constist only of two classes. Mathematically speaking, binary classification is the task of

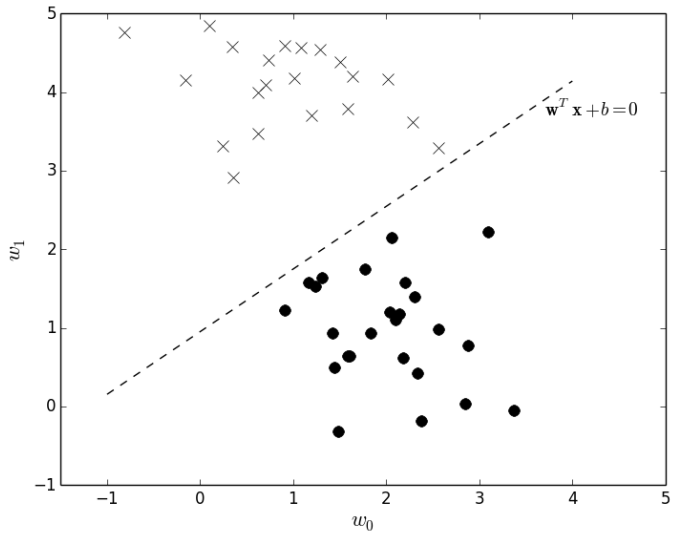


Figure 2: The binary classification task is to find $\mathbf{w}$ and b for the hyperplane that separates the two classes (crosses and dots). Here, two features are used.

finding a function, $f(\mathbf{x})$, that separates two classes, denoted by $y \in \{-1, +1\}$. Assuming the classes are linearly separable, this function is represented by a hyperplane

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \tag{1}$$

where $\mathbf{w}$ is the weight vector, b is a bias and $\mathbf{x}$ is a point in the feature space to be assigned a class. In a 2-dimensional space a hyperplane corresponds to a straight line where $\mathbf{w}$ is the slope and b is the point where the line crosses the y-axis. This is illustrated in Figure 2.

Assuming the two classes are linearly separable we seek to find a hyperplane such that for all observations, $\mathbf{x}_i$, for $i = 1, \ldots, N$.

$$\mathbf{w}^T\mathbf{x}_i + b \geq 0 \quad \text{for } y_i = +1 \tag{2}$$

$$\mathbf{w}^T\mathbf{x}_i + b < 0 \quad \text{for } y_i = -1 \tag{3}$$

For a multi-class problem the *one-vs-the-rest* procedure can be applied where a single binary classifier per class is trained using the class as positive label and all observations not in that class as negative label. The final label is decided by the model with highest confidence. Multi-class classification is illustrated in Figure 3 where the dotted lines represent each classifier trained on one class versus the rest and the filled areas denote the decision boundary. [20]

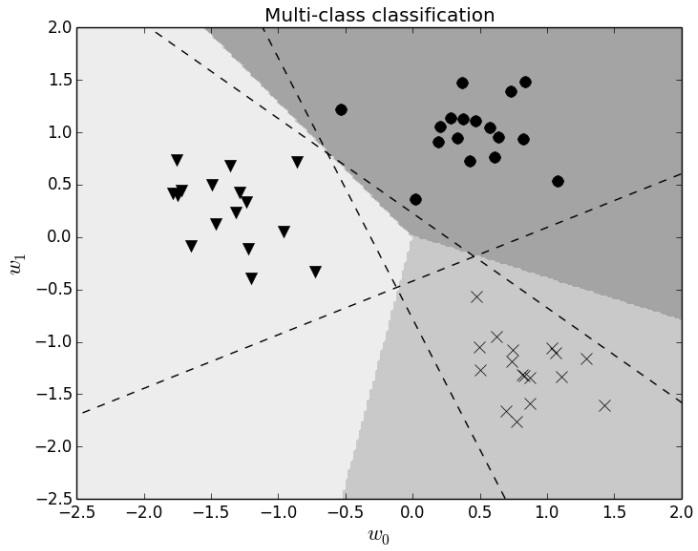


Figure 3: Multi-class classification where the filled areas denote the decision boundary and the dotted lines denote each one-vs-the-rest classifier. Here three classes (triangles, dots and crosses) are separated using two features.

As briefly mentioned in Section 2, *Support Vector Machines* (SVM) [21] are models for binary classification widely used for both linearly separable and non-separable classes. The objective of an SVM is to find a hyperplane that maximizes the margin between the two classes. There is not a single optimal hyperplane since scaling of $\mathbf{w}$ and b yields infinite solutions. By convention pick the so called *canonical hyperplane*

$$|\mathbf{w}^T\mathbf{x} + b| = 1 \tag{4}$$

where $\mathbf{x}$ is the training data closest to the separating hyperplane, known as *support vectors*, see Figure 4. For the case of linearly separable classes, finding a separating hyperplane can now be posed as the constrained problem of finding $\mathbf{w}$ such that

$$\mathbf{w}^T\mathbf{x}_i + b \geq +1 \quad \text{for} \quad y_i = +1 \quad \forall i \tag{5}$$

$$\mathbf{w}^T\mathbf{x}_i + b \leq -1 \quad \text{for} \quad y_i = -1 \quad \forall i \tag{6}$$

which can be combined into the equivalent expression

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i \tag{7}$$

The margin, M, is twice the distance from the separating hyperplane to one support vector and can be expressed, using (4), as

$$M = 2\frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|_2} = \frac{2}{\|\mathbf{w}\|_2}. \tag{8}$$

Maximizing the margin can be formulated as a constrained minimization problem

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \forall i \end{aligned} \tag{9}$$

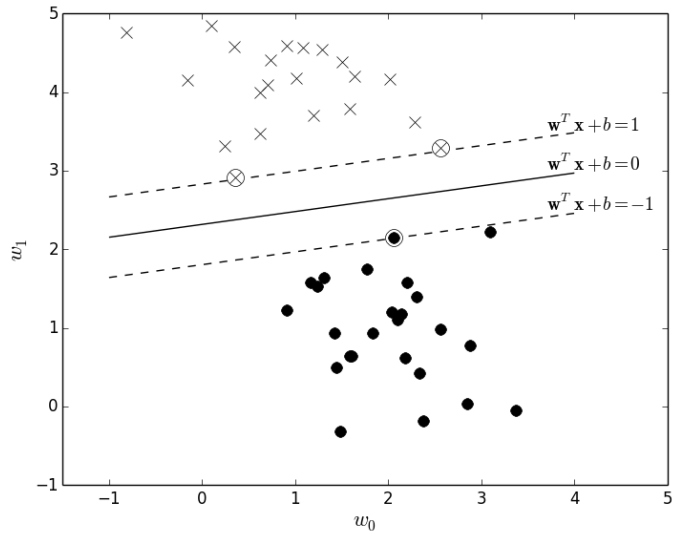


Figure 4: Illustration of SVM's margin maximization approach to the binary classification problem separating two classes (crosses and dots). Circled points denote support vectors.

where y_i is the label for sample $\mathbf{x}_i$. Equation (9) can be rewritten as an unconstrained minimization problem using Lagrangian multipliers as

$$\mathbf{w}^* = \arg\min_w \frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2. \tag{10}$$

where λ is a tuning parameter. This problem can be solved using, for example, a quasi-Newton method which is discussed more in detail in Section 4.5. For non-separable data an additional error term based on the distance from the separating hyperplane is added to the objective function and constraints.

4.3 Conditional Random Fields

A regular classifier, such as an SVM, classifies each token separately, i.e. assuming the tokens are independent of each other. However, in the natural language the meaning of a word or a sequence of words can differ depending on in what context they are used. For example, *Charles Bonnet* could be either a person's name or a syndrome, which is a kind of visual hallucinations experienced by a person suffering from partial or severe blindness. It is often obvious from the context. Hence, a way of modeling this context relation is motivated to improve the performance of a classification task.

One way to take the context into account for a specific token is to incorporate the labels of nearby tokens, i.e. taking the class of neighboring tokens into account when classifying the targeted token. This is the concept of *Conditional Random Fields* (CRF) introduced by Lafferty et al. [22] and briefly mentioned in Section 2. To define a CRF we need a set of real-valued *feature functions*, f_k. Generally, arguments of a feature function are a sentence, $\mathbf{x}$, the position, t, of a token in $\mathbf{x}$, the label, y_t, of the target token and the labels, $y_{j\neq t}$, of any of the tokens in the sentence.

A feature function describes one aspect of the context of the target word. For example, a feature function could indicate that, given that the previous token is "Dr.", the target word should be labeled as a person.

To describe the general form of a CRF we introduce the concept of *sequences* and *states* where, in this report, a sentence can be seen a sequence of tokens. Each token can be in a certain state corresponding to the PHI category of the token.

Incorporating the labels from arbitrary tokens in a sequence will lead to a complex and computationally heavy model. By instead taking only the previous label, y_{t-1}, into account, we are using the special case of a *linear-chain CRF*. This property of assuming that the current state only depends on the previous state is known as the *Markov property*. Below follows the general definition of a linear-chain CRF.

Definition 1. *For $k = 1, \ldots, K$, let $\lambda = \lambda_k$ be a parameter vector and $\{f_k(y_t, y_{t-1}, \mathbf{x}_t)\}_{k=1}^{K}$ be a set of real-valued feature functions where K is the total number of feature functions. Then the linear-chain CRF is defined as*

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t, t) \right\} \tag{11}$$

where $\mathbf{x}_t$ is the token at position t in a sequence $\mathbf{x} = x_1, \ldots, x_T$ with corresponding label y_t. Here it is assumed that $\mathbf{x}_t$ contains all components needed from the sequence $\mathbf{x}$ to compute features at time t, hence the vector notation. For example, if the next token x_{t+1} is used as a feature, $\mathbf{x}_t$ is assumed to include the identity of word x_{t+1}. $Z(\mathbf{x})$ is a normalization function specific for a sequence, $\mathbf{x}$, defined as

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp\left\{ \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t, t) \right\}. \tag{12}$$

A CRF is a powerful tool for sequential labeling because it can take arbitrary real-valued feature functions that can use any of the tokens, x_t, in the sequence, $\mathbf{x}$. Each feature function, f_k, is associated with a weight, λ_k, which can be interpreted as how much the feature function contributes to a certain label. Compared to an SVM, a CRF classifies a whole sentence while an SVM classifies each token in a sentence separately.

4.4 Finding the optimal path

When labeling unseen data the objective is to find the single best sequence of states, $\mathbf{y} = y_1, \ldots, y_T$, for a given sequence of observations, $\mathbf{x} = x_1, \ldots, x_T$, and the model parameters λ, i.e. maximize $p(\mathbf{y}|\mathbf{x}, \lambda)$ which is equivalent to maximizing $p(\mathbf{y}, \mathbf{x}|\lambda)$. This can be done efficiently by using the *Viterbi algorithm* [20].

The Viterbi algorithm is easier to understand if we represent the model as a lattice. Figure 5 shows a lattice of states and time steps where the solid blue line represents the global optimal path through the sequence, the dashed lines represents the optimal path in each state and the grayed out lines represents sub-optimal paths not saved in the Viterbi algorithm. At a specific time step, $t \in \{1, \ldots, T\}$, and state, $s \in S$, there are many paths arriving to the corresponding state. However, we only need to save the previous state

of the path with highest probability so far (dashed and solid colored lines). This means that, at each time step, t, we only need to store a total of $|S|$ paths, one for each state. At the final time step, T, the probabilities are compared. The final state with highest probability corresponds to the path with the overall highest probability (solid line). By simply backtracking that path, we obtain all labels in the sequence [20].

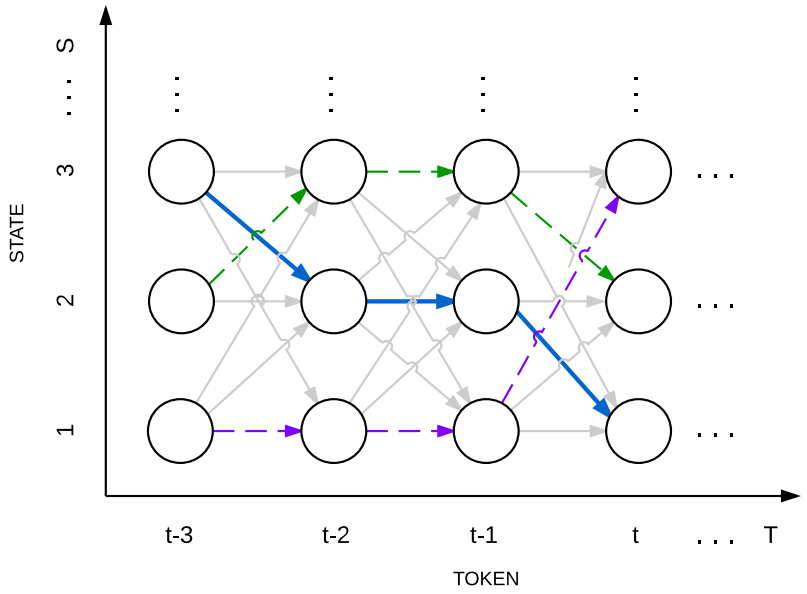


Figure 5: A lattice, representing the possible states as rows and the tokens as columns, illustrating the Viterbi algorithm.

Before presenting the formal definition of the Viterbi algorithm, we will first reformulate the definition of the CRF to be consistent with the literature. Definition 1 can be rewritten as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, \mathbf{x}_t), \tag{13}$$

where $\Psi_t(y_t, y_{t-1}, \mathbf{x}_t)$ is the transition probability from state y_{t-1} to y_t for an observation $\mathbf{x}_t$ and is defined as

$$\Psi_t(y_t, y_{t-1}, \mathbf{x}_t) = \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t, t) \right\} \tag{14}$$

We also need to define an expression for the most probable path for a partial sequence of observations

$$\delta_t(i) = \max_{y_1, y_2, \ldots, y_{t-1}} p(y_1, y_2, \ldots, y_t = s_i, x_1, x_2, \ldots, x_t|\lambda) \tag{15}$$

where s_i is the last state of the partial sequence $x_1, \ldots, x_t$. In [23] the Viterbi algorithm is explained by dividing the procedure into the four steps stated below:

1. *Initialization:* Initialize the most probable path for each state at the first token.

$$\delta_1(i) = \Psi_1(y_i, y_0, \mathbf{x}_1), \qquad 1 \leq i \leq N \tag{16}$$

$$\varphi_1(i) = 0. \tag{17}$$

2. *Recursion:* For each token, find the path arriving in state i with highest probability, $\delta_t(j)$. Also, save the previous state, $\varphi_t(j)$, of this path for future reference when backtracking the optimal path.

$$\delta_t(j) = \max_{1 \leq i \leq N} \big\{ \Psi_t(y_j, y_i, \mathbf{x}_t) \delta_{t-1}(i) \big\}, \qquad 2 \leq t \leq T \quad 1 \leq j \leq N \tag{18}$$

$$\varphi_t(j) = \arg \max_{1 \leq i \leq N} \big\{ \Psi_t(y_j, y_i, \mathbf{x}_t) \delta_{t-1}(i) \big\}, \qquad 2 \leq t \leq T \quad 1 \leq j \leq N. \tag{19}$$

3. *Termination:* At the final token in the sequence the path determine the state with highest probability. The saved path leading to this state is the optimum.

$$p^* = \max_{1 \leq i \leq N} \big\{ \delta_T(i) \big\}, \tag{20}$$

$$y_t^* = \arg \max_{1 \leq i \leq N} \big\{ \delta_T(i) \big\}. \tag{21}$$

4. *Path backtracking:* Backtrack through the saved states of the optimal path to retrieve the states for each token.

$$y_t^* = \varphi_{t+1}(y_{t+1}^*), \qquad t = T-1, \quad T-2, \ldots, 1. \tag{22}$$

In the following section we discuss how the weights associated with each feature can be found.

4.5 Optimization Algorithm

Finding the optimal weights for the separating hyperplane is stated as an optimization problem where we want to minimize the cost with respect to some constraints. The optimization algorithm used in this report is the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [24]. It is a widely used optimization algorithm to numerically find local maxima or minima of an objective function.

Recall from from basic calculus that both first and second derivatives are needed to determine the characteristics of an extreme value. L-BFGS is a member of the quasi-Newton methods family which are methods for finding extrema when the Jacobian (first derivative) or Hessian (second derivative) is unavailable or too expensive to compute. The L-BFGS algorithm approximates the inverse Hessian just like its parent, the BFGS method, but does not store the approximation as a dense $n \times n$ matrix in memory, where n is the number of variables. To explain why L-BFGS requires less memory we start by introducing the general quasi-Newton algorithm.

Let x_k be an approximate solution at iteration k, $f(x)$ be the objective function we want to minimize and $\nabla f(x_k)$ its gradient. Furthermore, define

$$s_k \equiv x_{k+1} - x_k \text{ and } y_k \equiv \nabla f(x_{k+1}) - \nabla f(x_k) \tag{23}$$

Also, let $H_k = B_k^{-1}$ be the inverse Hessian at time iteration k. The BFGS algorithm takes the following form.

Algorithm 1 Quasi-Newton algorithm

1: Specify initial guess of the solution x_0 and initial inverse Hessian approximation H_0.
2: **for** k=0,1, ... **do**
3: **if** $|\nabla f(x_k)| < \epsilon$ **then**
4: Optimization converged. Stop!
5: **end if**
6: Compute search direction $p_k = -H_k \nabla f(x_k)$
7: Use line search to determine $x_{k+1} = x_k + \alpha_k p_k$
8: Compute

$$s_k = x_{k+1} - x_k$$
$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

9: Update the inverse Hessian $H_{k+1} = H_k + \dots$
10: **end for**

Depending on what type of quasi-Newton method is used the inverse Hessian is updated differently. For the BFGS algorithm the Hessian is updated according to

$$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^t s_k}. \tag{24}$$

The Hessian update formula for B_k has an associated formula for updating the inverse Hessian, used in line 9 in Algorithm 1. The inverse Hessian update formula for BFGS is

$$H_{k+1} = \Big[I - \frac{s_k y_k^T}{y_k^T s_k}\Big] H_k \Big[I - \frac{y_k s_k^T}{y_k^T s_k}\Big] + \frac{s_k s_k^T}{y_k^T s_k} \tag{25}$$

$$= H_k - \frac{s_k (H_k y_k)^T + (H_k y_k) s_k^T}{y_k^T s_k} + \frac{y_k^T s_k + y_k^T H_k y_k}{(y_k^T s_k)^2} (s_k s_k^T). \tag{26}$$

Observing that $y_k^T H_k y_k$ and $y_k^T s_k$ are scalars, this expression can efficiently be computed without storing temporal matrices in memory. The L-BFGS method also exploits the fact that the next search direction $p_{k+1} = -H_{k+1} \nabla f(x_{k+1})$ can be computed by directly applying the inverse Hessian update formula:

$$p_{k+1} = -H_{k+1} \nabla f(x_{k+1}) \tag{27}$$

$$= \Big[I - \frac{s_k y_k^T}{y_k^T s_k}\Big] H_k \Big[I - \frac{y_k s_k^T}{y_k^T s_k}\Big] \nabla f(x_{k+1}) + \frac{s_k s_k^T}{y_k^T s_k} \nabla f(x_{k+1}) \tag{28}$$

and instead of storing H_k it can be defined by once again using the update formula in terms of y_{k-1}, s_{k-1} and H_{k-1}. H_{k-1} can then be defined in terms of y_{k-2}, s_{k-2} and H_{k-2}. The update formula can recursively be expanded r times which requires H_{k+1-r} to be initialized. Usually, H_{k+1-r} is set to be the identity matrix, I. Following this procedure one only need to store the sequences s_k and y_k in memory resulting in $2 \times r \times n$ elements instead of $n \times n$. [24]

4.6 Regularization

According to [20], when a statistical model is unnecessarily complex, such as using too many features for the number of observations, the problem of *overfitting* often occurs. Rather than learning the general pattern the model memorizes the data points presented to it. An overfitted model fails to describe the underlying pattern in the data and instead models the noise. The model's performance on the training set will be promising but when applied to new data it will show a poor performance.

Regularization is a powerful way to reduce overfitting by eliminating unimportant features and emphasizing the ones with information. Applying regularization is done by adding the norm of the weight vector to the objective function. This results in weights that tend to be as small as possible, reducing the complexity of the model. The norm of a vector can be represented in different ways. This report takes use of L_1 (29) and L_2 (30) regularization using the following norms:

$$L_1: \quad \|\mathbf{w}\|_1 = \sum_i |w_i| \tag{29}$$

$$L_2: \quad \frac{1}{2}\|\mathbf{w}\|_2^2 = \frac{1}{2}\sum_i w_i^2. \tag{30}$$

Using a linear combination of these two norms is called *Elastic net* regularization [25]. Applying regularization is as simple as adding a regularization term to the objective function we want to minimize:

$$\mathbf{w}^* = \arg\min_w \frac{1}{n}\sum_{i=1}^{n} f(y_i, \mathbf{w}^T\mathbf{x}_i) + C \cdot R(\mathbf{w}) \tag{31}$$

$$R(\mathbf{w}) = \lambda_1\|\mathbf{w}\|_1 + \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2 \tag{32}$$

where $\mathbf{w}^*$ denotes the optimal set of $\mathbf{w}^*$, $f(y_i, \mathbf{w}^T\mathbf{x}_i)$ is a model specific objective function, $R(\mathbf{w})$ is a regularization term with elastic net regularization, and C is the regularization strength. Usually $\lambda_1 = \alpha$ and $\lambda_2 = 1 - \alpha$ where α is referred to as the L_1 *ratio*. When $\alpha = 1$ the a full L_1 regularization is obtained, and the more commonly used L_2 regularization is obtained when $\alpha = 0$.

As seen in the contour plots in Figure 6, the elastic net has characteristics of both L_1 and L_2 penalties. The contour of the elastic net regularization has singularities in the vertices, just like the L_1 regularization, and is also strictly convex, like the L_2 regularization. Note also that L_1 regularization is convex but not strictly convex. The implications of L_1 regularization are that it yields a sparse solution where some coefficients are pushed down to exactly 0 while for L_2 regularization, the coefficients are pushed towards 0 but never reach it. A property of L_1 regularization is that because of its sparse solution it will serve as an automatic feature selector. This can be highly desirable when dealing with a large feature space. L_2 regularization has a *grouping effect* because of being strictly convex which makes highly correlated features vary their coefficients together in contrast to L_1 regularization which would select one of the correlated features and push the rest to 0. [25]

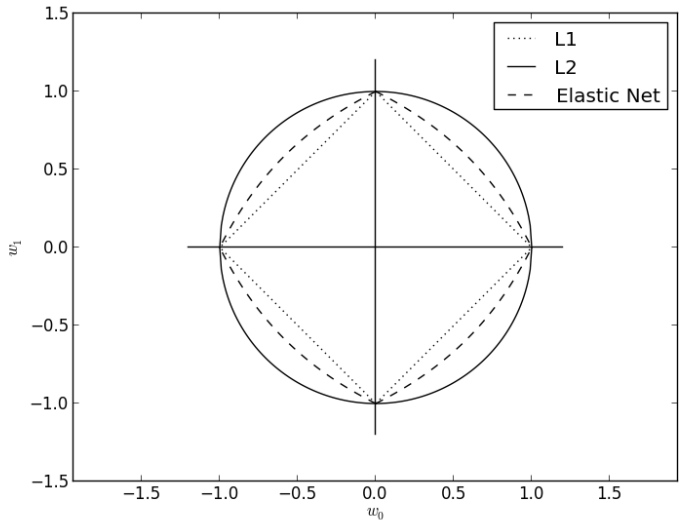


Figure 6: Contour plot for L_1, L_2 and Elastic net regularization.

5 Method and Materials

The solutions brought up in this report are assembled using a combination of pattern matching, dictionary lookups and machine learning techniques. UMC uses the Java-based Apache UIMA (Unstructured Information Management Architecture) framework for computational tasks in the area of Natural Language Processing (NLP), see Section 5.1. UIMA provides helpful tools for annotating text documents and accessing annotations in a convenient way. This framework is one of the main tools used when developing the de-identification algorithm.

The first step is to obtain a data set to use for training and evaluation. The data sets used in this report are described 5.2. The process for de-identifying narratives is outlined in Figure 7 and described in more detail in the following sections. The process can be divided into a training phase and a evaluation phase. The training phase starts with training documents getting pre-processed before features used by the statistical models are extracted. The extracted features are written to file which a model can read and use for training. Finally, the trained model is packaged to a convenient format.

The evaluation phase follows the same procedure as the training phase but rather than saving the features to a file the features are fed to the model just trained which outputs new annotations. Lastly, the evaluation documents are post-processed where sensitive entities are masked or modified.

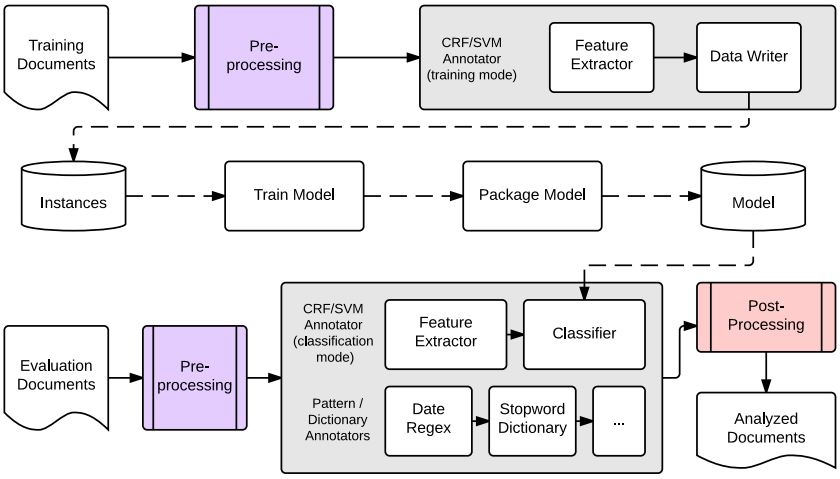


Figure 7: Outline of the process for de-identifying case narratives.

5.1 Unstructured Information Management Architecture

Narratives in case reports handled by the UMC contain a lot of unstructured information. An application for de-identifying narrative text would be required to analyze large volumes of unstructured information. The Apache Unstructured Information Management Architecture (UIMA) [26] framework is developed just for this task and is used in this thesis. In addition to unstructured text, UIMA can also be used to analyze unstructured information such as audio or video.

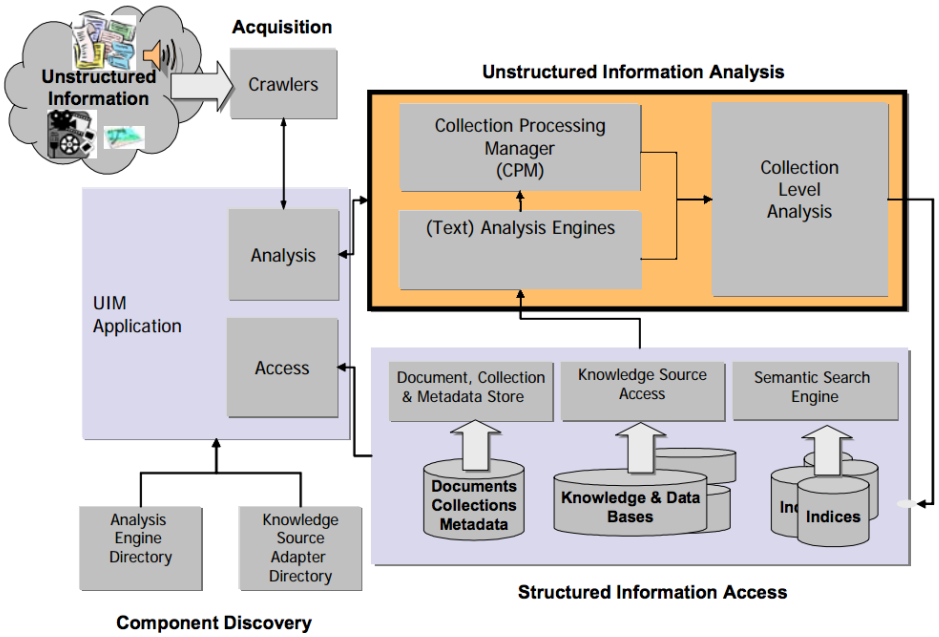


Figure 8: UIMA high-level architecture, as described in [26].

UIMA is a Java-based architecture which provides component interfaces, data representation and design patterns and enables a modular approach to help analyze unstructured

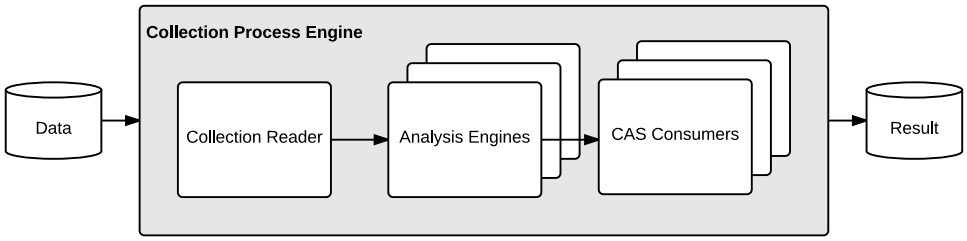


Figure 9: Conceptual view of building a analysis pipeline in UIMA.

data. The framework provides tools not only for processing local text documents but also for building semantic search engines, web services and cluster management for large scale computations. A high-level overview of UIMA's architecture is illustrated in Figure 8.

In this thesis, focus has mostly been on building components in the Unstructured Information Analysis area (top right part of Figure 8, marked in orange) that are combined into a pipeline, also called a *Collection Process Engine*. Components operate on a data structure called *Common Analysis Structure* (CAS) where objects and relations are stored. Objects labeling a text span in the CAS are called *Annotations* and can, for example, represent a token or an entity, e.g. a location or date. All annotations includes at least two features, namely *begin* and *end*, which for text documents refer to integer offsets in the document representing a span of text. UIMA provides tools to create user defined annotations for entities of interest where additional attributes can be added to the annotations. The components can be divided into three main categories:

- **Collection Readers** - Components that read from a data source and initiate the CAS.
- **Analysis Engines** - Components that modify existing or add new annotations or relations to the CAS.
- **CAS Consumers** - Final processing of the CAS, e.g. building a search index or populating a database.

An analysis engine may contain a single *annotator* (*Primitive AE*) or it may be composed of multiple annotators (*Aggregate AE*). Figure 9 shows a conceptual overview of how these components are combined. [26]

UIMA also provides a tool for visualizing annotations and manually annotating text documents and is useful when generating the gold standard data set.

There are several applications and libraries that use the UIMA architecture. One is the open-source toolkit ClearTK [27] which is used for developing statistical NLP components in the UIMA framework. The toolkit provides useful interfaces for evaluating models using cross validation and common measurement calculations. ClearTK also contains wrappers for different statistical models such as Conditional Random Fields, Support Vector Machines and Maximum Entropy. For the CRF an implementation by Naoaki Okazaki, CRFsuite [28], was used. The LIBLINEAR package by Rong-En Fan et. al [29] was used as the implementation of the SVM. Both provide a fast and scalable

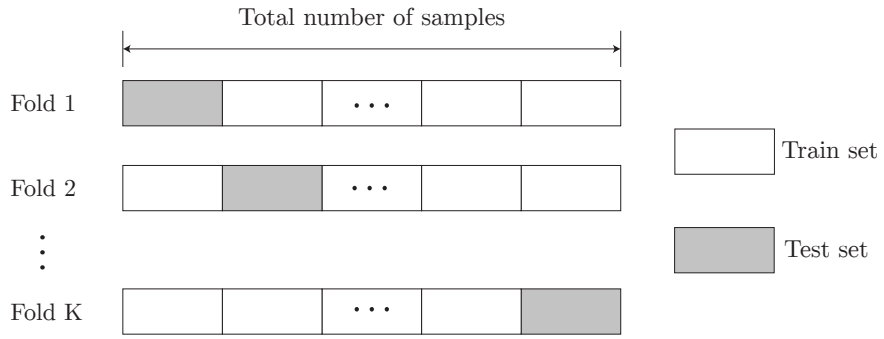


Figure 10: Illustration of K-fold cross validation.

implementation with highly customizable training settings.

Apache clinical Text Analysis and Knowledge Extraction System (cTAKES) [30] is built upon the UIMA architecture and is focused on information extraction from electronic health records. cTAKES provides multiple components, such as sentence detection, tokenization and POS tagging, trained and adapted for clinical documents. It also includes named-entity recognition of medications, diseases, symptoms, anatomical sites and procedures based on dictionary lookup in the Unified Medical Language System (UMLS) dictionaries SNOMED CT® [31] and RxNorm [32]. Some of cTAKES's components use the statistical tools provided by ClearTK.

5.2 Data sets

A manually annotated data set had to be created for training and evaluation, a so called *Gold Standard*. This included extraction of reports from VigiBase and manually annotating the sensitive information of the reports. Manual annotation can be a tedious task because each token needs to be assigned a class and requires human verification since the meaning of the word can differ depending on context. A high variety of documents are desired to get a model as general as possible and to reduce overfitting. Increasing the number of documents in the data set quickly leads to a vast amount of tokens to manually annotate. A K-fold cross validation is a way of using all data for both training and testing where each observation is used for testing only once. By dividing the data set into K folds it is possible to train K different models using K-1 folds as training set and test on the remaining fold as illustrated in Figure 10. The final performance is the average over all folds. The purpose of using K-fold cross validation is to estimate the performance of a model when a test set is not available. As most of the articles described in [14] the performance is measured using precision, recall and F-measure. A 5-fold cross validation was applied to the training set when developing the features, patterns and parameters used by the models.

5.2.1 VigiBase

The initial data set consisted of 100 randomly selected reports from UMC's case report database VigiBase. A medical doctor in pharmacovigilance at the UMC assisted in determining what should be included and excluded in the different categories of sensitive information. The reports were manually annotated in the UIMA CAS Editor which allows a user to select spans of a text document and tag the spans with an annotation.

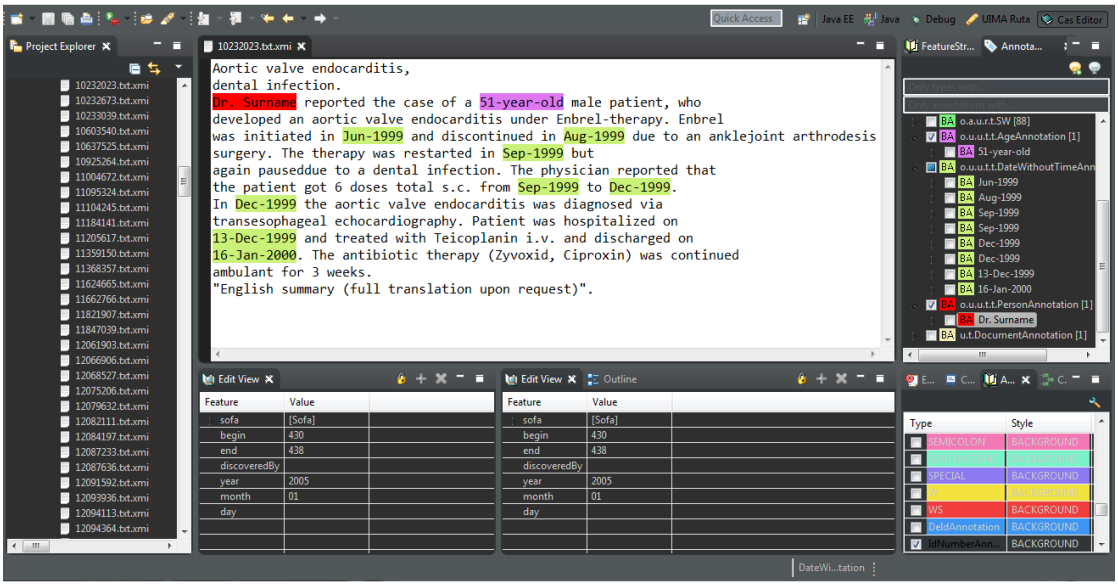


Figure 11: The graphical interface used to visualize annotation and manually annotate case reports. Names, dates and ages have been altered to protect the identity of the patient.

Figure 11 shows the graphical user interface of the CAS Editor. Tokens not tagged with any of the sensitive categories are interpreted as *Outside* tokens. When it was concluded that the 100 reports contained too few instances of some categories, the data set was extended to include 300 reports. In addition, an evaluation set of 100 reports was manually annotated.

The reports are not restricted to a specific country as long as the text is in English, in order to get a data set representing the current situation at the UMC. Figure 12 shows the most frequent countries reporting in English where it is obvious that the majority of reports comes from the U.S. followed by India. English is widely used in the Indian education system and is necessary for working with medicine [33]. Problems originating from English not being the native language are therefore unlikely to be a major issue.

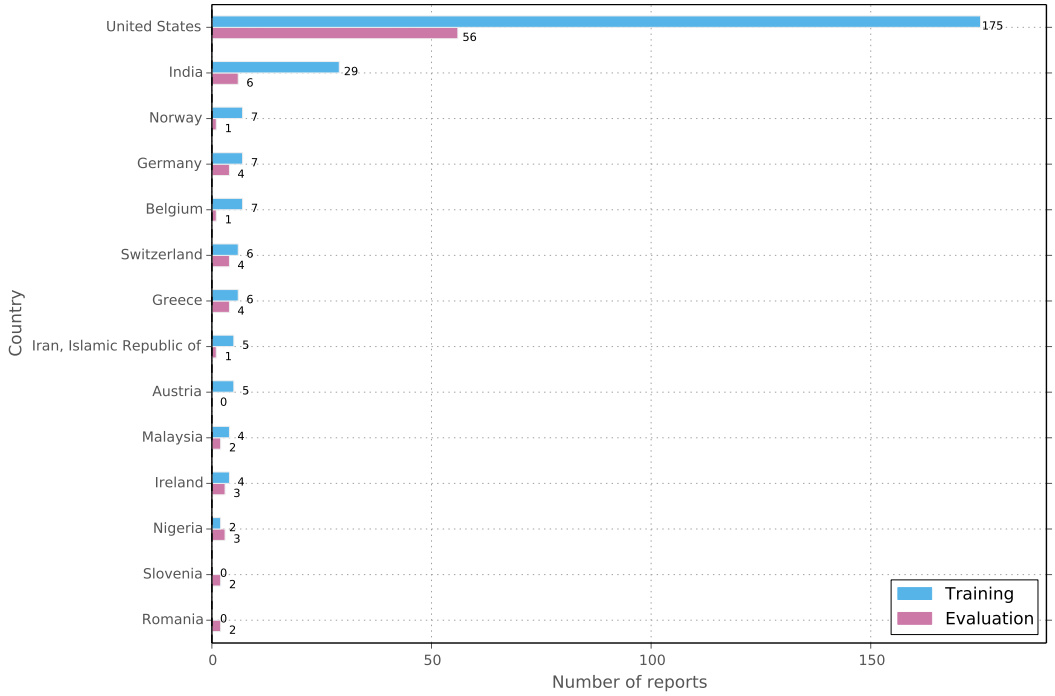


Figure 12: Distribution of reports over the most frequent countries for train and evaluation set.

The distribution of the final data set for training and evaluation can be seen in Table 1. As seen, there are plenty of data for dates and ages but the amount of names and organizations are small. With that amount of observations it is difficult to find a general pattern for names and organizations.

Table 1: Training (300 reports) and evaluation (100 reports) set distribution of gold standard annotations.

Category	Training	Evaluation
Date	553	185
Age	109	30
Location	25	9
Organization	5	2
Person	4	0
Total	696	226

In addition to obvious date references, e.g. *"24 September 2014"*, less direct dates are also included in the *Date* category, such as common holidays (Thanksgiving, Christmas, etc.) and expressions like *"first week of graduate school"* since they refer to a small interval or a specific point in time. These expressions are included since a sentence like *"Delays because of Thanksgiving traffic resulted in death of the patient."* would implicitly specify the date of death which is counted as sensitive information. It should be noted that this sort of expression rarely occurs in the VigiBase reports. Entities of relative time references like "two weeks later" were not annotated since without the reference point it is impossible to identify the actual date.

The four entities found in the category *Person* consist of one initial, one patient first name and two doctor names, where one of the doctor names had a typo making the name and a location being put together. This is an example of a common issue in these reports. Often the narratives are written in the form of a note with incorrect grammar, typos and abbreviations which can be troublesome for tokenizers and POS taggers trained on properly written documents.

Organizations consist mainly of specific hospitals that could be linked to a specific location. These entities were included since specific hospitals can pinpoint a certain location connected to the patient. Other organizations, e.g. pharmaceutical companies, are not included since they are not related to the individual.

In some reports ID numbers were found but not related to the individual. The numbers mainly referred to already encrypted identifiers from other reports or studies, and batch numbers of drugs related to the production of the medicine. These were therefore excluded from the *Id Number* category.

5.2.2 i2b2 as benchmark

The i2b2 data set consists of a training and test set containing 669 and 220 documents respectively. The training data set was reduced to 100 training documents to reduce the execution time of the procedure of finding optimal regularization strength. The category distribution is shown in Table 2.

Table 2: Distribution over categories for the full training (669 reports) and evaluation (220 reports) set of the i2b2 data set as well as reduced version of the sets.

Category	**Training**	**Test**	**Reduced train**
Date	5167	1931	717
ID	3666	1143	550
Person	3365	1315	522
Organization	1724	676	268
Phone	174	58	25
Location	144	119	15
Age	13	3	3
Total	14253	5245	2100

Important to note here is that even though the i2b2 data set comes from the area of medicine, the i2b2 reports do not have the same format nor follows the same annotation procedure as the VigiBase reports. In the i2b2 data set, *Dates* consist *only* of days and months, this is in contrast to the VigiBase data set where both full dates and years only where marked as dates. Furthermore, the i2b2 *Age* category contains only ages when they exceed 89 years and all forms of *ID Numbers* are annotated. For consistency with the VigiBase annotations, *Doctors* and *Patients* were mapped to the more general *Person* category. Literature about automatic de-identification frequently use the i2b2 de-identification challenge data set to train and evaluate models [14]. For our purposes, it can provide a benchmark and show how our model compares to others and perhaps more importantly more data to be used in combination with VigiBase data.

5.3 Evaluation measures

Each token is assigned a class indicating if it is sensitive or not. Because there are usually more words that are not sensitive, this leads to an imbalanced class distribution. The conventional *accuracy*, defined as the fraction of all observations correctly classified, is not well suited for this situation. A simple example is shown in 5.1. [34].

Example 5.1. *Suppose we have a data set containing observations from two different classes with the proportion 99% majority class and 1% minority class. Using accuracy on this data set and classifying all majority class observations correctly and all minority class observations incorrectly will give an accuracy of 99%. This is easily deceived as a really good performance. But if the goal was to predict the minority as well as possible this classification is worthless.*

Another way to measure performance that takes class imbalance into account is precision, recall and F_1-score where precision and recall is calculated from the confusion matrix,

		Predicted label	
		−	+
Actual label	−	True Negative (TN)	False Positive (FP)
	+	False Negative (FN)	True Positive (TP)

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{33}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{34}$$

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{35}$$

The F_1-score is a special case of the F_β-score, with $\beta = 1$, where the F_β-score is the weighted harmonic mean of precision and recall defined as

$$F_\beta = (1 + \beta^2)\frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{36}$$

For Example 5.1 we would have a recall of 0 resulting in an F_1-score of 0 which is a more adequate measure than the accuracy of 99%.

A *text span* is represented by a start and an end offset in the document. To count a classification of a token as correct we can require that both the span and the category

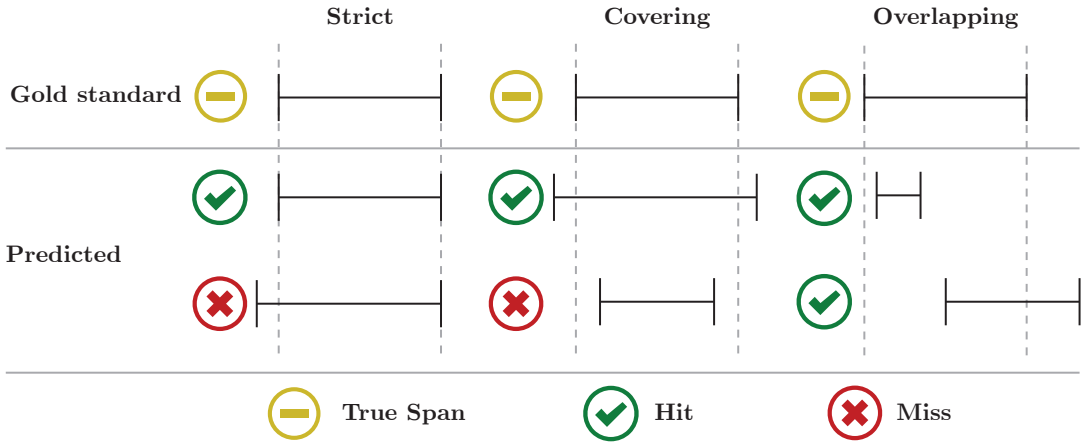


Figure 13: Illustration of the difference between the evaluation methods.

match the gold standard. However, the models make a classification of a single token and will be dependent on the tokenization of a sentence. If the predicted annotation includes a punctuation that is not present in the gold standard, the prediction will be counted as incorrect, even though the span covers the full gold span. To resolve this issue we introduce a more relaxed criterion where, as before, the category has to match, but the gold standard span only needs to be fully covered by the predicted span. This criterion will measure a model's ability to identify named entities and be certain that the whole gold span is covered. These criteria will be referred to as *Covering criteria*. Relaxing the covering criteria a little further by only requiring the spans to have any kind of overlap, we get an indication of how well the model can identify approximate locations of entities. We will refer to these criteria as *Overlapping criteria*. Figure 13 and Example 5.2 illustrate the differences of strict, covering and overlapping criteria.

Example 5.2.

Gold Standard
Gold*:* Patient has been sore since the age of **[91 years]**$_{gold}$, when she tripped and fell.

Strict Criteria
Hit: Patient has been sore since the age of **[91 years]**$_{pred}$, when she tripped and fell.
Miss: Patient has been sore since the age of **[91 years,]**$_{pred}$ when she tripped and fell.

Covering Criteria
Hit: Patient has been sore since the **[age of 91 years,]**$_{pred}$ when she tripped and fell.
Miss: Patient has been sore since the age of **[91]**$_{pred}$ years, when she tripped and fell.

Overlapping Criteria
Hit: Patient has been sore since the **[age of 91]**$_{pred}$ years when she tripped and fell.
Miss: Patient has been sore since the **[age of]**$_{pred}$ 91 years, when she tripped and fell.

5.4 Pre-processing

In this thesis, the *IOB2* format [9] is used to define the available states of a sequence. All tokens *outside* a PHI element are labeled with *O*. The token that *begins* a PHI element are labeled with *B-k* where *k* is the PHI category and the succeeding tokens are labeled *I-k*, being *inside* the PHI element. A concrete example is shown in Table 3.

Table 3: IOB2 format example.

Words	IOB2-tag
John	B-PERSON
Smith	I-PERSON
experienced	O
symptoms	O
January	B-DATE
20th	I-DATE

The models used in this thesis are based on classifying each token in a sentence. For that to be possible one first needs to identify sentences in a text segment and individual tokens in sentences. This is normally not as trivial as saying that a dot finishes a sentence or a whitespace separates tokens. Decimal numbers, hyphens, parenthesis and typos are some special cases that can complicate the task of defining sentences and tokens.

cTAKES [30] provides a sentence detector and tokenizer trained on a variety of documents. Before applying these on the text some simple cleanup is made which includes removal of contiguous whitespace and replacement of characters like $\&, >$ and $<$ to their word representation. A POS tagger is then applied to the text that adds a POS attribute to each token and with a chunker, noun and verb phrases are identified. These chunks are used in a dictionary lookup to the UMLS (SNOMED CT and RxNorm) dictionaries that contains clinical terminology and normalized names for clinical drugs respectively. The identified entities are later used as features to the classifiers.

5.5 Feature Engineering

In this chapter we describe how the regular expressions were chosen and lists the different expressions used. Additionally, we show and explain the features used for the statistical models.

5.5.1 Patterns for RegEx model

Regular expressions for the RegEx model has been constructed by the cyclic process illustrated in Figure 14. By visual inspection of the training data, certain age and date patterns were prominent. After creating a regular expression to cover the patterns, the model was applied on the training data to obtain the new performance. Visual inspection was then applied again to see what occurrences of ages and dates that were missed. This was repeated until the performance was satisfiable and the most frequent patterns were covered. Outliers, such as entities containing typos, were not covered by regular expressions. A complete

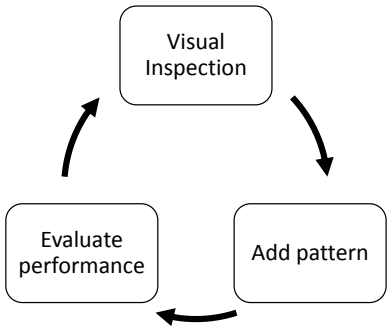


Figure 14: The process of constructing regular expressions.

list of the regular expressions used, as well as examples for each pattern, is shown in Table 4-7.

Table 4: Date RegEx patterns

Variable Name	RegEx pattern
MONTH_STR	`(jan(uary)?|feb(ruary)?|mar(ch)?|apr(il)?|may|june?|july?|aug(ust)?|sep(t|tember)?|oct(ober)?|nov(ember)?|dec(ember)?)`
MONTH_2_DIG	`(0[1-9]|1[0-2])`
MONTH_1OR2_DIG	`(0?[1-9]|1[0-2])`
YEAR_2TO4_DIG	`((19|20)?\d{2})`
YEAR_4_DIG	`(((19)|(20))\d{2})`
DAY_2_DIG	`(([0-2]\d)|(3[01]))`
DAY_1OR2_DIG	`(((0?|[12])\d)|(3[01]))`

Table 5: Date RegEx patterns combined

RegEx pattern	Example
`{DAY_1OR2_DIG}\W?{MONTH_STR}\W?{YEAR_2TO4_DIG}`	12Jan2014, 1-december-15
`\b{DAY_2_DIG}\W?{MONTH_2_DIG}\W?{YEAR_2TO4_DIG}\b`	12012014, 31-12-15
`\b{DAY_1OR2_DIGs}\W{MONTH_1OR2_DIG}\W{YEAR_2TO4_DIG}\b`	12/1/2014, 6/12/15
`\b{YEAR_2TO4_DIG}\W?{MONTH_2_DIG}\W?{DAY_2_DIG}\b`	2014-01-12
`{MONTH_STR}\W?{DAY_1OR2_DIG}\W\W?{YEAR_2TO4_DIG}`	January 24, 2014
`\b{MONTH_2_DIG}\W?{DAY_2_DIG}\W?{YEAR_2TO4_DIG}\b`	01/12/14 (American format)
`\b{MONTH_1OR2_DIG}\W{DAY_1OR2_DIG}\W{YEAR_2TO4_DIG}\b`	1/8/14 (American format)
`{MONTH_STR}\W?{YEAR_2TO4_DIG}`	jan2014, december 2015
`{YEAR_4_DIG}\W?{MONTH_STR}`	2014 oct
`\b{MONTH_2_DIG}\W{YEAR_2TO4_DIG}\b`	01/14
`\b{YEAR_4_DIG}\b`	2004

Table 6: Age RegEx patterns

Variable Name	RegEx pattern
TIME_UNIT	`(years?|months?|weeks?)`
VALUE	`(\d{1,2}(\d|(\W\d))?)`
VALUE_STR	`(zero|one|two|three|...|twenty)`

Table 7: Age RegEx patterns combined

RegEx pattern	Example
`({VALUE}|{VALUE_STR}).(y\.?o\.?|{TIME_UNIT}.?(old))`	eleven months old, 9 y.o.
`(aged?( of)?|a\.?o\.?).{VALUE}`	aged 32, age of 41, a.o. 85

These regular expressions cover most of the patterns observed in the training set where the entities missed in the training set consist of typos and abnormalities and was not taken into account for in the expressions developed. Trying to cover all typos and abnormalities would lead to overfitted regular expressions since we are describing the noise instead of the underlying pattern.

5.5.2 Features for statistical models

Aramaki et al. [9] found that PHI elements often occur at the beginning or end of a document. To determine if this is the case for VigiBase data a feature function was developed to measure the position of a token in a narrative text. The position is a value in the interval $[0, 1]$ where 0 is the first token of the text and 1 is the last. Figure 15 is a box plot showing how the token position varies between categories. One can see that ages have a tendency to occur at the beginning of documents while dates are distributed more uniformly. Additionally, Figure 16 shows the distribution of a token's position in a sentence; note that ages appears more often in the first half of a sentence than in the second half.

Due to the small amount of observations of locations, organizations and persons it is not possible to draw a general conclusion regarding these categories. Figure 15 & 16 indicate, however, that these uncommon categories are found in the beginning of a report where persons are often positioned in the beginning of a sentence and locations are found at the end of a sentence.

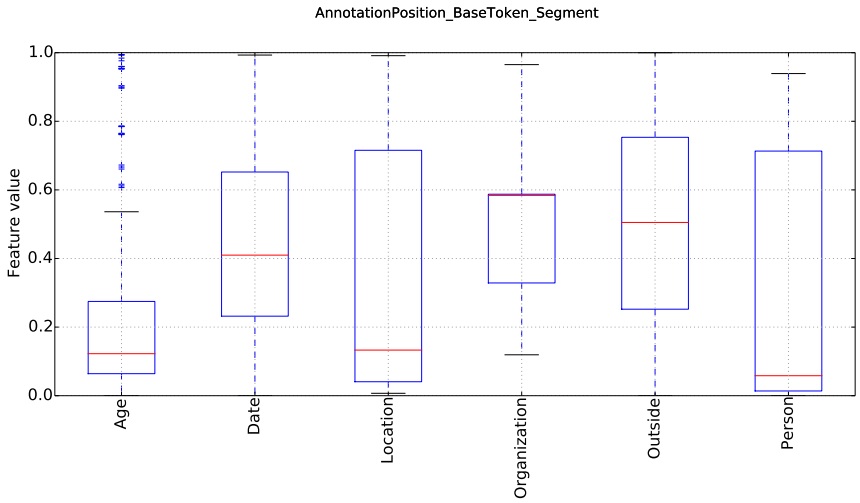


Figure 15: Box plot of the positions of tokens in a narrative divided by category.

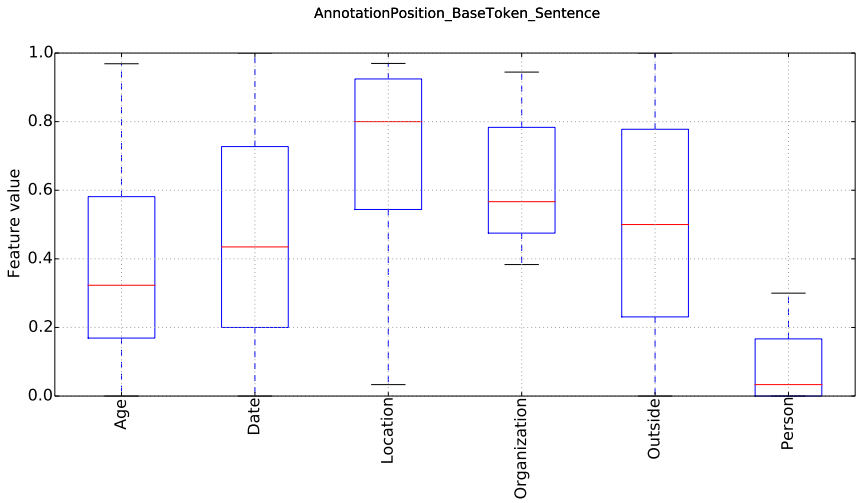


Figure 16: Box plot of the positions of tokens in a narrative divided by category.

With the insight that the token position could help in determining a token's category it would make sense to use the token position as a feature in our models.

In this thesis, we use local features, non-local features and dictionary lookups from external sources. Before the features are generated, narratives are annotated by lookup of tokens and phrases in external sources. The external sources are the UMLS dictionaries for medical terms and drugs, RxNorm and SNOMED CT. A binary feature is generated if the token is part of an annotation generated by an external source. A complete collection of feature is listed in Table 8. Below follows a description of the non-obvious features:

Capital Type describes the capitalization of a token. In practice, the feature function produces one out of four possible binary variable which indicates the capitalization type of a token: `ALL_UPPERCASE`, `ALL_LOWER_CASE`, `INITIAL_UPPERCASE` or `MIXED_CASE`.

Numerical Type indicates if a token contains digits and if so tries to describe in what setup they occur. Just like for the capital type, this feature function creates a binary feature for a suitable numerical type. The possible numerical types a token can have are `ALL_DIGITS`, `YEAR_DIGIT` (4 digits starting with 1, 20 or 21), `ALPHANUMERIC`, `SOME_DIGITS` (Some digits and non-letters) and `ROMAN_NUMERAL`.

N-gram suffix of a token is the last N characters of the token. For the experiments 2-gram and 3-gram suffixes are used.

Character Pattern describes the pattern based on the Unicode General categories [35] of each character in a token. A complete list of character types is listed in Appendix B. A simplified pattern is used where consecutive character types are merged to one instance of that type, e.g. the text `Ab01` with character pattern `LuLlNdNd` would be reduced to `LuLlNd`.

Lists of countries, common holidays, stopwords, weekdays and months are stored in text files. Sign/Symptoms, Disease/Disorder, Medication, Anatomical Site and Laboratory procedures annotations are generated by cTAKES dictionary lookup to UMLS. Person title annotations are created using the regular expression `\b((Dr|Mr|Mrs|Ms|Sr|Jr)\.?|Miss|Phd)\b`.

Table 8: Features used by both CRF and SVM.

Local features	**External sources**	**Non-local features**
Target Token (TT)	Countries	Token position in Sentence
TT in lowercase	Common Holidays	Token position in Document
Capital Type of TT	Stopwords	Sentence position in Document
Numeric Type of TT	Weekday	
TT contains hyphen	Month	
TT 2 & 3 gram suffix	Person titles	
TT Character Pattern	Sign/Symptom	
POS Tag of TT	Disease/Disorder	
TT text length	Medication	
TT contains X,Y or Z?	Anatomical Site	
	Laboratory procedure	
Above features for 3 preceding and following tokens		

5.6 Regularization

In the training procedure of a statistical model a regularization strength, C, has to be determined. By using a 5-fold cross validation with the training documents as data set and with multiple values of C the optimal regularization strength is found. When optimal C is determined, all training documents are used to train the model and the performance is measured by applying the model on the evaluation set.

Just like the regularization strength constant C affects the performance, finding an optimal L_1-ratio, α, can also improve the results. Using the optimal C found from previous cross validation and once again applying 5-fold cross validation on the training set with varying α we get the F_1-score, precision and recall as a function of α. After picking the optimal α all training documents are used to train the statistical model which is then evaluated on the evaluation set.

5.7 Varying amount of medical records

We would like to know if 300 documents are enough to get a desirable performance or if adding more data would improve the result. Therefore, an experiment was set up to show the performance of the classifiers when increasing amount of available data. The model was trained repeatedly by randomly selecting a proportion of the 300 available training documents, using the regularization strength C found as described in Section 5.6. Starting at 30 documents (10%) and incrementing with 30 documents, after 100 repetitions we can calculate the average and spread of the score. The procedure is outlined in Algorithm 2.

Algorithm 2 Varying number of available documents

Let D be all 300 documents available for training
Let E be all 100 documents available for evaluation
Let p be a proportion of training documents to keep for training
Let $r = 100$ be the number of repetitions for each p
for each $p = 0.1, 0.2, \ldots, 1.0$ **do**
    for r times **do**
        Let $D_p \subset D$ be a random proportion p of D
        Let M be a model trained on D_p
        Apply M to E and evaluate by computing and storing $\mathbf{F}_1$-score, **P**recision and **R**ecall
    end for
    Calculate average score, $\bar{F}_p, \bar{P}_p, \bar{R}_p$ and standard deviation, $\sigma_p^F, \sigma_p^P, \sigma_p^R$
end for
Plot score and deviation as a function of p

5.8 Post-processing

After identifying sensitive entities the final step of de-identifying a narrative text is to replace the entities found with an informative substitute that differs depending on the category of the entity. The logic for each category was implemented in what is here referred to as a *masker*. If multiple models are applied there may be multiple annotations overlapping for the same entity, hence a way of choosing which annotation to mask needs to developed.

For the case when annotations are overlapping the annotation coming from the model with highest covering precision was used since it is the most probable option to cover all sensitive information. Applying all three models evaluated in this report the decision order would be

1. RegEx
2. CRF

3. SVM

If desired, this decision order could be made category-specific. This could make a small improvement but would increase the complexity of the process making it more difficult to maintain.

Recall that we want to de-identify dates by adding a random offset, but to be able to add an offset to a date, we must first identify what components (day, month, year) of a date are present. Since there is no unified date format in the narratives this task can be complicated. The approach used in this thesis is the following. Each date annotation is mapped to one or several date formats defined by the RegEx patterns mentioned in 5.5.1. For a document, the formats can be sorted by frequency, with ambiguous dates parsed using the most frequent format. The procedure is illustrated in Table 9. Successfully parsed dates will have a document-specific random offset, from an administrator configurable interval, added and presented in the de-identified narrative in an unambiguous format with the same resolution as in the original text.

Table 9: Illustration of the process of selecting date formats.

Available formats	Dates found	Possible formats	Format frequency	Final format
A : MM-dd-YYYY	03-04-2014	[A,B]	B : 2	B
B : dd-MM-YYYY	16-05-2014	[B]	A : 1	B
C : MMM-YYYY	Jan 2014	[C]	C : 1	C

Age annotations are first scanned for consecutive digits and if found are converted to an integer. The unit of the integer is assumed to be in years if no sign of *day, week* or *month* is present. If the integer is greater than a user specified threshold the whole annotation will be replaced with `[AGE > {THRESHOLD}]`.

For categories with unimplemented maskers the substitute is `[{CATEGORY}]`. Entities that can not be parsed to a specific format are replaced by a substitute like the one from an unimplemented masker. This ensures that as much sensitive information as possible is removed from the narrative. After de-identification, the same narrative as shown in Figure 11 looks like the following:

> Aortic valve endocarditis, dental infection. [PERSON] reported the case of a 41-year-old male patient, who developed an aortic valve endocarditis under Enbrel-therapy. Enbrel was initiated in [FEBRUARY 2005] and discontinued in [APRIL 2005] due to an anklejoint arthrodesis surgery. The therapy was restarted in [MAY 2005] but again pauseddue to a dental infection. The physician reported that the patient got 6 doses total s.c. from [MAY 2005] to [AUGUST 2005]. In [AUGUST 2005] the aortic valve endocarditis was diagnosed via transesophageal echocardiography. Patient was hospitalized on [13 AUGUST 2005] and treated with Teicoplanin i.v. and discharged on [21 AUGUST 2005]. The antibiotic therapy (Zyvoxid, Ciproxin) was continued ambulant for 3 weeks. "English summary (full translation upon request)".

6 Result

We describe the result in terms of precision, recall and $F-1$-score, both the overall performance of all categories as well as each category individually. With the result we can compare our models and determine what model is suitable under what circumstances.

6.1 Regular Expressions versus Statistical Models

First, the regularization strength, C, had to be picked. This was done by using a 5-fold cross validation. Most important is to find all sensitive information, which is why C was picked with respect to optimizing recall. Figure 17 shows how the recall varies with C. Best performance was obtained when $C_{CRF} = 10^{-4}$ and $C_{SVM} = 10^{-1}$ for CRF and SVM respectively. A CRF and SVM were then trained on the full training set using optimal C. The classification performance on the evaluation set over all categories is displayed in Table 10-12.

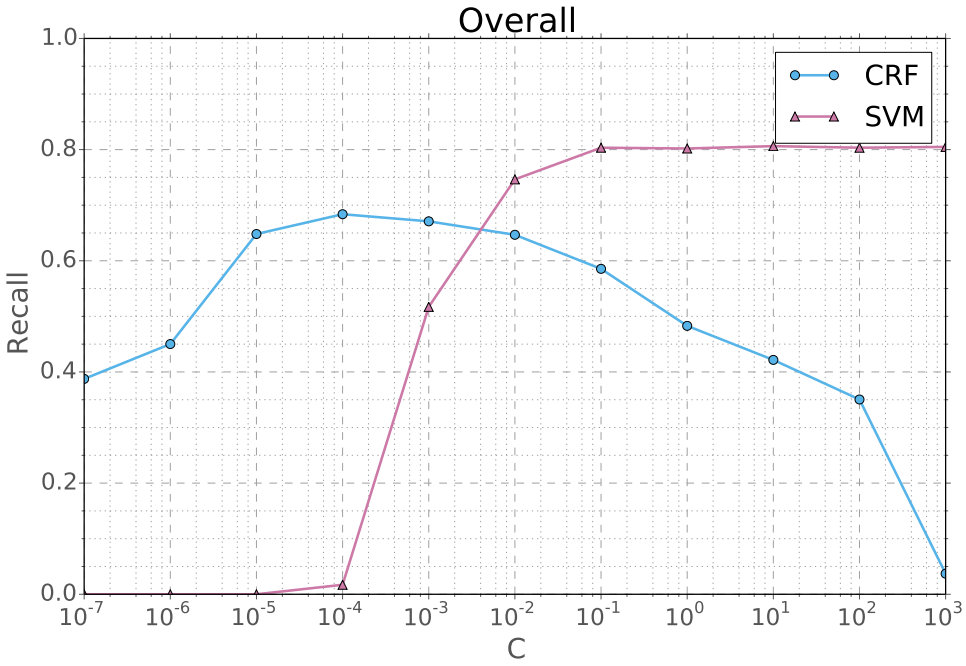


Figure 17: Recall when varying the regularization factor C.

Table 10: Performance using RegEx patterns on the evaluation set containing 100 reports.

Strict			**Covering**			**Overlapping**			**Category**
P	R	F_1	P	R	F_1	P	R	F_1	
0,925	0,876	0,900	0,930	0,881	0,905	0,949	0,898	0,923	OVERALL
0,923	0,800	0,857	0,923	0,800	0,857	1,000	0,867	0,929	Age
0,930	0,941	0,935	0,936	0,946	0,941	0,947	0,957	0,952	Date
1,000	0,000	0,000	1,000	0,000	0,000	1,000	0,000	0,000	Location
1,000	0,000	0,000	1,000	0,000	0,000	1,000	0,000	0,000	Organization

Table 11: Performance using CRF on the evaluation set containing 100 reports.

Strict			**Covering**			**Overlapping**			**Category**
P	R	F_1	P	R	F_1	P	R	F_1	
0,850	0,774	0,810	0,859	0,783	0,819	0,942	0,858	0,898	OVERALL
1,000	0,833	0,909	1,000	0,833	0,909	1,000	0,833	0,909	Age
0,829	0,811	0,820	0,840	0,822	0,831	0,934	0,914	0,923	Date
1,000	0,000	0,000	1,000	0,000	0,000	1,000	0,000	0,000	Location
1,000	0,000	0,000	1,000	0,000	0,000	1,000	0,000	0,000	Organization

Table 12: Performance using SVM on the evaluation set containing 100 reports.

Strict			**Covering**			**Overlapping**			**Category**
P	R	F_1	P	R	F_1	P	R	F_1	
0,901	0,885	0,893	0,905	0,889	0,897	0,923	0,907	0,915	OVERALL
1,000	0,933	0,966	1,000	0,933	0,966	1,000	0,933	0,966	Age
0,885	0,919	0,902	0,891	0,924	0,907	0,911	0,946	0,928	Date
1,000	0,222	0,364	1,000	0,222	0,364	1,000	0,222	0,364	Location
1,000	0,000	0,000	1,000	0,000	0,000	1,000	0,000	0,000	Organization

Comparing Table 10-12 we see that RegEx patterns has a superior overall F_1 performance using all three evaluation criteria. However, the SVM is in the same range with less than 1 percentage point in F_1 score below RegEx for all criteria. Looking at the categories separately, we see that SVM has a superior performance for *Ages* but lacks in performance for *Dates* compared to RegEx.

Another finding is the 8 percentage point jump in overall F_1 score for the CRF going from covering to overlapping criteria. This is the result of having problems classifying the inside tokens of an entity but success in finding the first token.

We can look at the missed or incorrectly classified entities individually to dig deeper into the reason for the unsuccessful classification. One concrete example is the following text from one report.

... history hepatitis.XXOn 19-Jul- 2011, the patient started ...

Because of the whitespace after the hyphen in the date, this entity is not found with the RegEx rule but was found using either the CRF or SVM.

RegEx produces false positives for numbers that have the same format as a date, like the example below which was identified by the RegEx patterns as a date but not by the statistical models:

... related to rotigotine.XXAssociated linked case no: 050368

Furthermore, the RegEx patterns used does not handle intervals, like in the following case:

... cases occurred on October-November 2013. Viogen has stopped ...

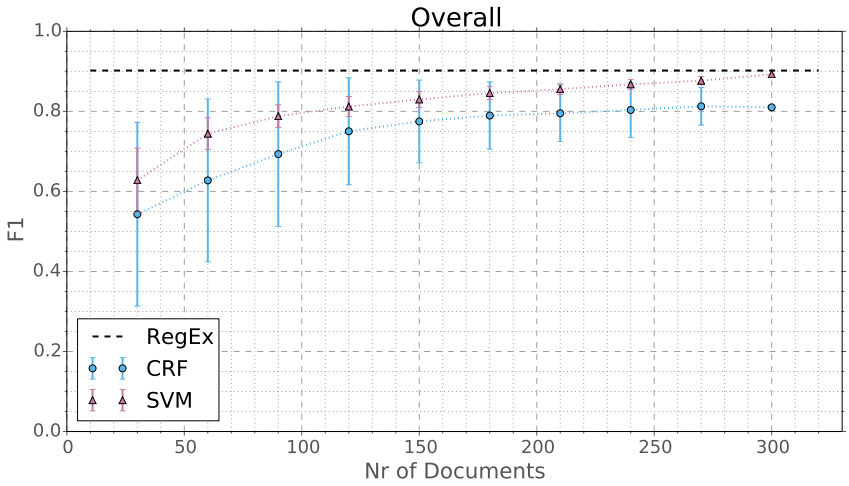
In this case the RegEx pattern annotates *November 2013*, while for both statistical models the whole interval is annotated.

6.2 Varying amount of medical records

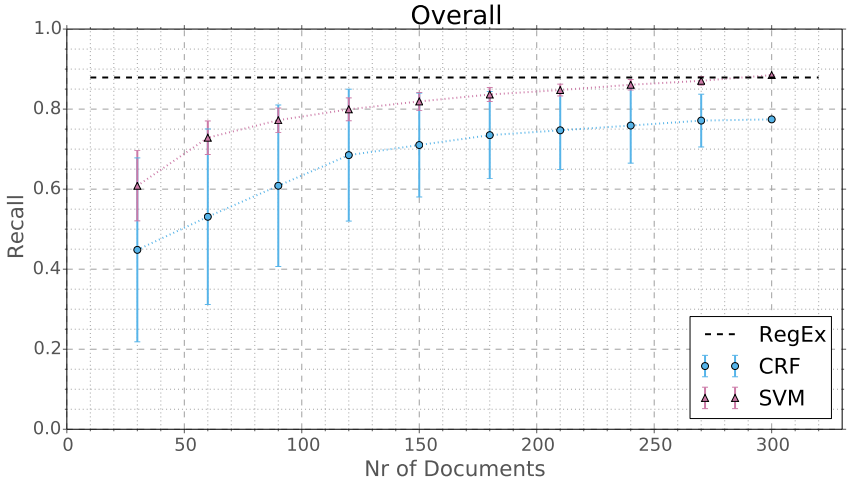
Figure 18 shows how the F_1-score, precision and recall over all categories vary with increasing number of training documents for both CRF and SVM. It can be seen that the SVM outperforms the CRF because of SVM's superior recall. It is, however, worth noticing that the CRF has a better precision on average than the SVM for a low number of reports.

The SVM is more robust compared to the CRF because of its small standard deviation, though it should be noted that there exists a bias in the deviations. Since there is a limit of 300 reports in the training simple combinatorics says that there are more ways to pick 30 reports than 270 out of a total of 300 reports. With an increasing number of reports it is more likely that the same report is used for training in several repetitions, reducing the deviation of performance.

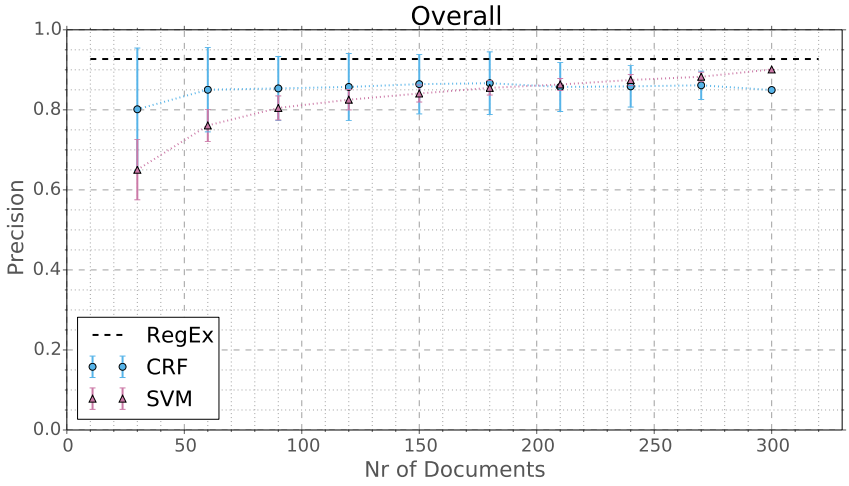
Looking at the trends in overall covering recall, Figure 18b, one can not tell if they have reached a plateau yet. It is not obvious if more data would increase the performance or not. The result of each category separately can be found in Appendix C.



(a) Overall F_1-score, covering criteria.



(b) Overall recall, covering criteria.



(c) Overall precision, covering criteria.

Figure 18: Results when varying amount of available records.

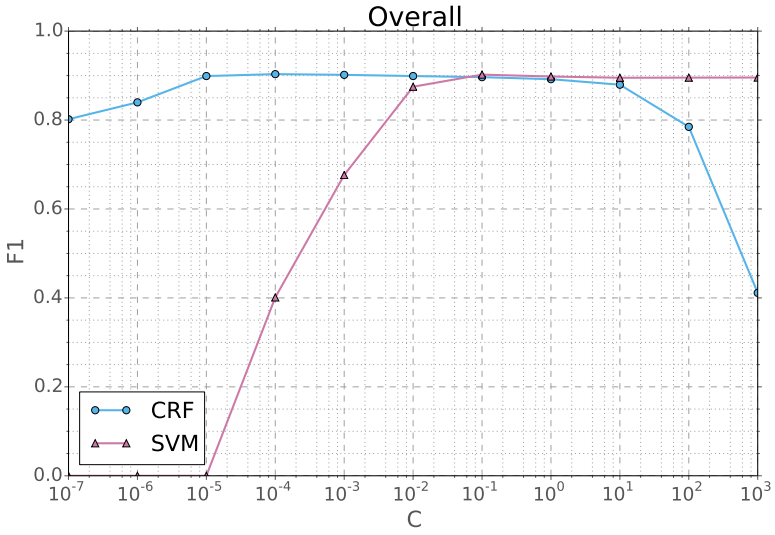


Figure 19: F_1 performance of CRF and SVM on a reduced I2B2 data set as a function of regularization strength, C.

6.3 i2b2 challenge as benchmark

Regularization strength was determined by applying a 5-fold cross validation to the reduced i2b2 training set for varying C. The result is shown in Figure 19 where the optimal values can be found to be $C_{CRF} = 10^{-4}$ and $C_{SVM} = 10^{-1}$. With these values of C a CRF and a SVM model are trained on all documents in the full training set and applied to the full evaluation set. The final performance is presented in Table 13 and 14. Because of the difference in format and annotation procedure between i2b2 and VigiBase data the regular expressions developed for the VigiBase data is not applicable to the i2b2 data and is therefore left out in this section.

Table 13: Performance using CRF on the i2b2 evaluation set, where regularization strength, $C = 10^{-4}$

Strict			**Covering**			**Overlapping**			**Category**
P	R	F_1	P	R	F_1	P	R	F_1	
0.950	0.877	0.912	0.953	0.880	0.915	0.960	0.887	0.922	OVERALL
1.000	0.333	0.500	1.000	0.333	0.500	1.000	0.333	0.500	Age
0.926	0.969	0.947	0.928	0.972	0.949	0.929	0.972	0.950	Date
0.960	0.993	0.976	0.961	0.994	0.977	0.962	0.995	0.978	IdNumber
0.857	0.151	0.257	0.857	0.151	0.257	1.000	0.176	0.300	Location
0.988	0.503	0.667	0.988	0.503	0.667	1.000	0.509	0.675	Organization
0.969	0.918	0.943	0.976	0.924	0.949	0.997	0.944	0.970	Person
0.968	0.517	0.674	0.968	0.517	0.674	1.000	0.534	0.697	Phone

Table 14: Performance using SVM on the i2b2 evaluation set, where regularization strength, $C = 10^{-1}$

Strict			**Covering**			**Overlapping**			**Category**
P	R	F_1	P	R	F_1	P	R	F_1	
0.918	0.915	0.917	0.921	0.918	0.920	0.970	0.967	0.968	OVERALL
1.000	0.000	0.000	1.000	0.000	0.000	1.000	0.000	0.000	Age
0.962	0.964	0.963	0.963	0.965	0.964	0.983	0.985	0.984	Date
0.988	0.992	0.990	0.988	0.992	0.990	0.991	0.996	0.993	IdNumber
0.493	0.303	0.375	0.493	0.303	0.375	0.863	0.529	0.656	Location
0.704	0.750	0.726	0.714	0.760	0.736	0.886	0.944	0.914	Organization
0.938	0.925	0.931	0.944	0.931	0.938	0.985	0.971	0.978	Person
0.818	0.776	0.796	0.818	0.776	0.796	0.964	0.914	0.938	Phone

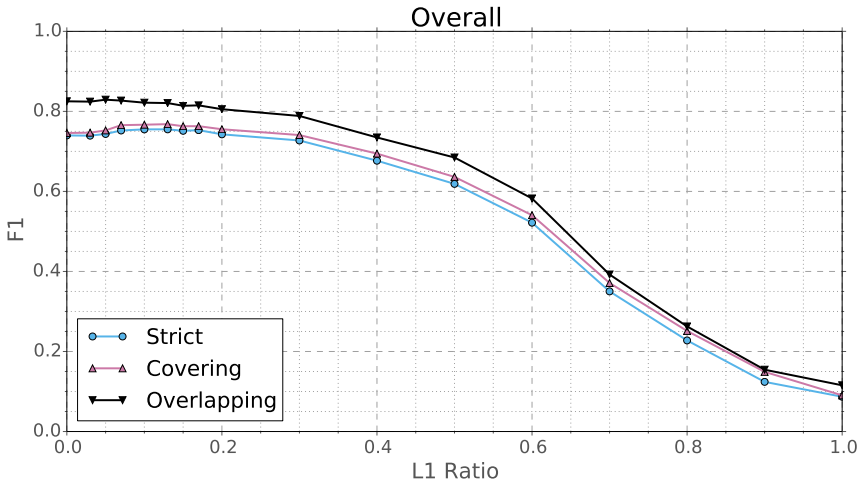
The results presented here, especially for the SVM, are comparable with the proposed machine learning methods in the 2006 i2b2 de-identification challenge [14]. The proposed models had an overall strict precision, recall and F_1-score all in the range 0.86-0.96 which both the CRF and SVM presented here also lie within. The models participating in the challenge had a superior performance in the categories with a low amount of data, such as locations, ages and phones.

The SVM achieves a good performance for organizations, which was not the case using VigiBase data. Both for i2b2 and VigiBase the organization category mainly contains entities of hospitals. We could therefore use a model trained on i2b2 data to classify organizations in VigiBase data. Applying the SVM model trained on i2b2 data to the VigiBase evaluation set the two existing organization entities was identified with overlapping criteria, though at the price of nine false positives.

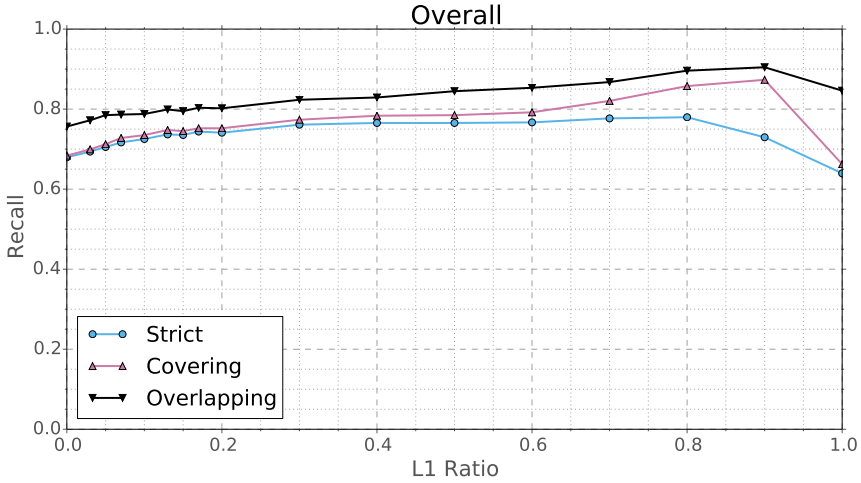
The VigiBase data set had all its age entities annotated with the intent of identifying the specific age in the post-processing, masking only the ones above a threshold. With the same reasoning as above a VigiBase model could help identifying ages in the i2b2 data set. Using the VigiBase-trained SVM from Section 6.1 and applying it to the i2b2 evaluation set two out of three ages over 90 are found with covering criteria. However, most of the other ages are found as well resulting in a huge amount of false positives. The high amount of false positives would be drastically reduced by removing age annotations with an age below 90.

6.4 Regularized CRF training

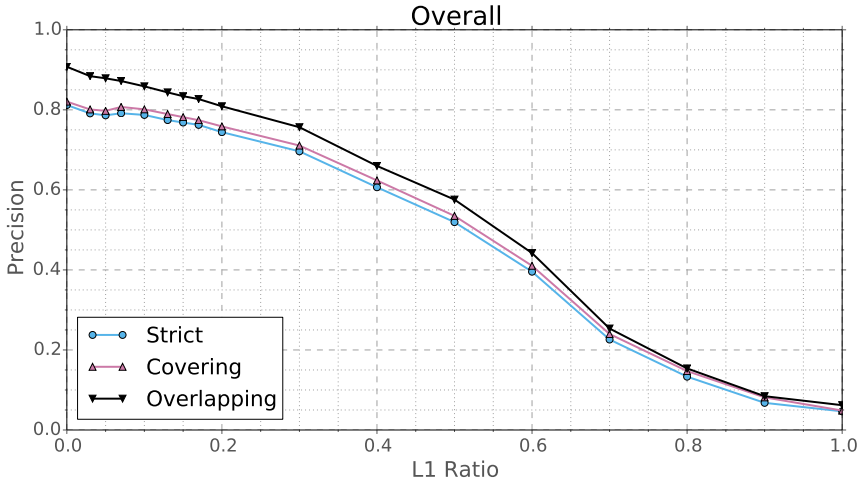
Using the optimal regularization strength, C, found in Section 6.1 and applying 5-fold cross validation on the training set with varying α we get the F_1-score, precision and recall as a function of α, shown in Figure 20. Optimal L_1-ratio for the covering criteria is obtained at $\alpha^* = 0.13$ when $C_{CRF} = 10^{-4}$. Applying the model, trained with α^*, to the evaluation set we get the result presented in Table 15. Comparing this with the CRF's result in Table 11, Section 6.1 we see a major increase in performance, especially for *strict* and *covering* evaluation criteria where the overall F_1-score has increased with just over 7 percentage points. Comparing the results with the SVM, Table 11, Section



(a) F_1 performance for CRF with varying L_1-ratio.



(b) Recall performance for CRF with varying L_1-ratio.



(c) Precision performance for CRF with varying L_1-ratio.

Figure 20: Overall performance of the CRF with varying α. $C_{CRF} = 10^{-4}$.

6.1, we see that CRF with elastic net regularization is at a competing performance. The difference is that the SVM has a slightly higher recall than the CRF, while the CRF has slightly higher precision, making the final F_1-score almost equal.

Table 15: Performance on evaluation set using an L_1-ratio, $\alpha^* = 0.13$ and $C_{CRF} = 10^{-4}$.

Strict			**Covering**			**Overlapping**			**Category**
P	R	F_1	P	R	F_1	P	R	F_1	
0,903	0,863	0,882	0,912	0,872	0,891	0,935	0,894	0,914	OVERALL
1,000	0,933	0,966	1,000	0,933	0,966	1,000	0,933	0,966	Age
0,888	0,903	0,895	0,899	0,914	0,906	0,926	0,941	0,933	Date
1,000	0,000	0,000	1,000	0,000	0,000	1,000	0,000	0,000	Location
1,000	0,000	0,000	1,000	0,000	0,000	1,000	0,000	0,000	Organization

6.5 Feature ranking

The coefficient of a feature can be interpreted as the feature's importance in deciding the class of a token. With a positive coefficient, an increasing feature value increases the likelihood of a token being of the targeted class. With a negative coefficient value, an increasing feature decreases the likelihood.

The features describing the annotation position, inspired by Aramaki et al [9], were proven to be successful for the VigiBase data as well. Both the CRF and the SVM had the feature AnnotationPosition_BaseToken_Sentence with a negative coefficient in the top 10 of classifying the first token of an *Age*. It means that a token is less likely to be the start of an *Age* entity if it is located at the end of a sentence. Otherwise, the top features for the *Age* category included, not surprisingly, features derived from the surrounding tokens being the words "years" or "old" or an hyphen. For *Dates* the most prominent features involved tokens containing numbers, a slash ('/') or ended with *er*, as in *Novemb**er*** or *Septemb**er***. Also, if the preceding token was *in* or *on* it was a good indication of the current token being the beginning of a *Date* entity.

For the i2b2 data set, looking at the coefficients of the models for the Person category the features with the highest absolute value are the Character Pattern `LuPo` (with a positive value) and `Medication` (with a negative value). The character pattern indicates that a token all in upper case and ending with a dot is likely to be a name while if the token is part of a medication entity it is unlikely to be a person's name.

A complete list of top features of all categories and data sets for both CRF and SVM can be found in Appendix D. The features are picked from the optimal model setup found in the experiments of this report.

7 Discussion

In the following sections we discuss the results obtained in the previous sections. We will compare the performance and discuss advantages and disadvantages of each model.

7.1 Regular Expressions versus Statistical Models

The result in 6.1 show that the RegEx had a better overall performance than the statistical models but that the SVM was superior for the *Age* category. With this information it would be possible to get the best out of two worlds by applying different models on different categories: SVM for ages and RegEx for dates.

We also saw clear examples of the disadvantages of RegEx where typos resulted in a missed entity and where a number had the same format as a date. The same examples showed the advantages of the statistical models. Because of the CRF and SVM being context aware, they did not classify this number as a date.

Most of the date and age entities missed by the RegEx patterns in the evaluation set could, with not too much effort, be covered by adding a few more patterns. Why these were not covered from the beginning is because the new patterns were not prominent in the training set and therefore not thought of when constructing the patterns. While RegEx patterns are often quick to implement and easy to test, a major drawback is that it will only find what is already known and will not contribute to any further understanding in how patterns emerge. While writing a pattern to find a date format might be trivial it is not obvious how to distinguish a person's name from a drug using RegEx. Statistical models on the other hand, can adapt when new data is presented and is more accepting to deviations in the usual patterns. Nevertheless, statistical models can require a vast amount of data, it takes time to train them and they can be more difficult to communicate to an audience without proper mathematical background. A summary of the pros and cons of regular expressions and statistical models are listed in Table 16.

Table 16: A summary of pros and cons for regular expressions and statistical models.

RegEx		**Statistical Models**	
Pros	Cons	Pros	Cons
Quick and simple to implement	Does not adapt to new data	Robust, can handle typos	The need of data
No training needed	Need to know all patterns in beforehand	Can adapt to new data	Can be complex and difficult to communicate
Easy to communicate	Can only find what is already known	Can find new insights	Training takes time

7.2 Varying amount of medical records

From the results of these experiments the low amount of data is an issue for not obtaining good performance. There are several ways to handle this issue where the most obvious

one is to get more data. Since the process of manually annotating text is time consuming one can try to find already annotated data from external sources with a format similar to VigiBase data. This was one of the reasons why the i2b2 data set was obtained.

Other ways to handle the issue of limited data is to assign class weights to each category to make an incorrect classification more costly or using over/under sampling techniques. Incorrectly classifying a token from a minority class would yield a greater impact on the change of coefficients in the training procedure than incorrectly classifying a token from a common class. Oversampling the minority classes, using a technique such as SMOTE [36], would create synthetic data points for the classes with low amount of observations reducing the class imbalance. This could help avoiding an overfitted model. One can also do more extensive feature engineering, developing hand crafted, informative features, specific for the data set. This typically requires thorough analysis and deep knowledge of the origin of the data.

7.3 i2b2 as benchmark

As seen in Section 6.3 the models developed in this report are performing at a level not far from the state of the art solutions. When applying a model trained on i2b2 data to the VigiBase evaluation set we managed to find, with overlapping criteria, the two *Organizations* which usually were missed when training on VigiBase data. And when doing it vice versa, applying VigiBase model on i2b2 data, we identify two out of three ages above 90 that were missed when the models where trained on i2b2 data only. This finding shows that data obtained from external sources could be used to improve the performance on the data set in focus.

7.4 Regularized CRF training

The improvement in performance comes from the feature selection in the L_1 property. There exist plenty of unnecessary and highly correlated features that are completely removed, resulting in a reduced feature space. Originally, almost 86000 features are active. Using elastic net regularization reduces this feature space to just over 8200 features, a reduction of more than 90%. In Figure 20 it is interesting that the recall increases with increasing α and reaches its peak at $\alpha = 0.9$ for covering criteria. The precision, however, is extremely low for increasing α. This could be an effect of adjusting the regularization strength C for the new L_1-ratio. This would require further analysis with an exhaustive search for each combination of α and C.

8 Conclusions

In this thesis, focus has been on the narrative field in the case reports handled at the UMC. First of all it can be stated that the narratives from VigiBase already are well de-identified for all categories except dates and ages compared to reports presented in the publicly available i2b2 de-identification data set. Because of the small amount of data, it was difficult to provide a statistical model for entities in uncommon categories. It was, however, possible to develop a statistical model for ages and dates that performed at the same rate or better compared to hand crafted regular expressions. The expressions could, however, easily be extended to achieve top performance for both ages and dates for the VigiBase reports. For the task of identifying ages and dates regular expressions would be sufficient, but to identify entities from categories like persons, organizations and locations more advanced methods, such as SVM and CRF, are required.

With optimal parameters the covering F_1 performance of the CRF (0.891) is very close to the SVM (0.897). The difference is that the CRF performs at a higher precision but lower recall than the SVM, i.e. SVM identifies more of the sensitive entities to the cost of more false positives. Finding all entities of sensitive character is crucial to completely de-identify a narrative text. Therefore the SVM is preferable over the CRF with the assumption that the training and evaluation samples are representative to all English reports in VigiBase.

The disadvantage of regular expressions was emphasized when switching from the VigiBase data set to i2b2. Because of the different formatting and manual annotation procedure the regular expressions developed for VigiBase data was of no use. New expressions would have to be created to match the patterns in the new data in contrast to the statistical models that could use the same feature functions for the new data. This shows the flexibility of statistical models.

From the experiments performed in this thesis we see that the CRF requires more data than the SVM to perform at the same level. The CRF is designed to handle structures in the data, i.e. the order of tokens in a sentence matters in the classification of the sentence. Narrative text in VigiBase reports contains a mix of medical notes without grammatical structure as well as grammatically correct written text. This may lead to inconsistency in the CRF training resulting in a decreased performance.

Some medications or diseases, such as Parkinson's disease, could easily be interpreted as a person's name (which Parkinson's disease actually originates from). Even though medications and diseases are not sensitive information, identifying them can help the statistical model to not classifying these type of entities as names.

It is also worth mentioning that parameter selection plays an important part in optimizing performance which can be seen from the experiments when varying the regularization strength parameter, C. and L_1-ratio, α. Additional parameter configuration to the L-BFGS method could improve the performance, such as tuning of stopping criteria, line search parameters and number of times to apply the Hessian approximation.

9 Future Work

In addition to narrative information the VigiBase reports also include structured information. Two subsections in the structural part lists suspected and concomitant drugs and potentially the start and stop dates of these drugs. De-identified narratives will have original dates replaced by a date with a random offset and the narratives often describe the different events of a treatment, such as when a drug was given or withdrawn. Because of this, the structured information causes a risk of re-identifying processed documents by matching drugs and dates from the structured fields with corresponding context in the narrative. Doing this it may be possible to derive the offset for all date and thereby convert all de-identified dates back to the originals.

One potential way of solving this problem is to apply more extensive analysis in the form of semantic role labeling where dates would be linked to medications and events. These events could then be extracted from the narrative and presented as structured information where only the time interval of the event is mentioned. If all critical information in the narrative could be extracted to a structured format, there is no need to present the original narrative. By not presenting the narratives the risk of distributing sensitive information would be reduced to a minimum.

With the i2b2 data being more dense of sensitive entities it would be possible to train statistical models using a data set mixed with VigiBase and i2b2 data. Just like when finding the L_1-ratio we could define a parameter indicating the proportion of i2b2 documents to include with the VigiBase reports and analyze how introducing i2b2 data affects the performance.

Combining VigiBase and i2b2 data would increase the amount of text processed resulting in longer execution times. Instead of using L-BFGS as the optimization algorithm one can use Stochastic Gradient Descent (SGD) [37] which allows *online learning*. In contrast to batch learning, like L-BFGS, SGD approximate the gradient of the objective function by considering a single element or a small batch at a time. The drawback is that there are more configuration parameters that need to be set.

Medication identification proved helpful when determining person names for the i2b2 data. Instead of using dictionary lookups to UMLS this could be replaced by the UMC Drug Dictionary which has been developed using VigiBase data and would be more accurate finding entities of drugs and medications, which could further help when identifying person names.

Some symptoms or diseases are more common in different seasons, such as the flu which occurs in the cold season of the year. When masking dates a random offset is added to each successfully parsed date without respect of season. We can instead add three random offsets, one for day, month and year respectively, where the interval of the random numbers is reduced to a small interval. This will modify the date but keep the approximate time of year, making the narrative less confusing.

References

[1] Marie Lindquist. VigiBase, the WHO global ICSR database system: basic facts. *Drug Information Journal*, 42(5):409–419, 2008.

[2] Ghazaleh Karimi, Kristina Star, Marie Lindquist, and Ralph Edwards. Clinical and scientific letters [letter to the editor]. *Clinical Medicine*, 14(3):326–328, 2014.

[3] HIPAA. HIPAA 'Protected Health Information': What does PHI include? `http://www.hipaa.com/2009/09/hipaa-protected-health-information-what-does-phi-include/`, September 2014.

[4] Beatrice Santorini. Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). 1990.

[5] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

[6] Association for Computational Linguistics. Pos tagging (state of the art). `http://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)`, November 2014.

[7] Kristina Toutanova and Christopher D Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70. Association for Computational Linguistics, 2000.

[8] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[9] Eiji Aramaki, Takeshi Imai, Kengo Miyo, and Kazuhiko Ohe. Automatic deidentification by using sentence features and label consistency. In *i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data*, pages 10–11, 2006.

[10] Rob Koeling. Chunking with maximum entropy models. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 139–141. Lisbon, Portugal, 2000.

[11] Xavier Carreras, Lluís Màrquez, and Lluís Padró. A simple named entity extractor using AdaBoost. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 152–155, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[12] Dekang Lin and Xiaoyun Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the*

4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, pages 1030–1038. Association for Computational Linguistics, 2009.

[13] Clete A Kushida, Deborah A Nichols, Rik Jadrnicek, Ric Miller, James K Walsh, and Kara Griffin. Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies. *Medical care*, 50:S82–S101, 2012.

[14] Stephane Meystre, F Friedlin, Brett South, Shuying Shen, and Matthew Samore. Automatic de-identification of textual documents in the electronic health record: a review of recent research. *BMC Medical Research Methodology*, 10(1):70, 2010.

[15] Shawn N Murphy, Griffin Weber, Michael Mendis, Vivian Gainer, Henry C Chueh, Susanne Churchill, and Isaac Kohane. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17(2):124–130, 2010.

[16] Wendy W Chapman, Prakash M Nadkarni, Lynette Hirschman, Leonard W D'Avolio, Guergana K Savova, and Ozlem Uzuner. Overcoming barriers to NLP for clinical text: the role of shared tasks and the need for additional creative solutions. *Journal of the American Medical Informatics Association*, 18(5):540–543, 2011.

[17] Oscar Ferrández, Brett R South, Shuying Shen, F Jeffrey Friedlin, Matthew H Samore, and Stéphane M Meystre. BoB, a best-of-breed automated text de-identification system for VHA clinical documents. *Journal of the American Medical Informatics Association*, 20(1):77–83, 2013.

[18] Dilip Gupta, Melissa Saul, and John Gilbertson. Evaluation of a deidentification (De-Id) software engine to share pathology reports and clinical documents for research. *American journal of clinical pathology*, 121(2):176–186, 2004.

[19] Jan Goyvaerts. Regular expressions. `http://www.regular-expressions.info/`, January 2015.

[20] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[21] Christopher J.C. Burges. A tutorial on Support Vector Machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[22] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[23] Lawrence Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[24] Igor Griva, Stephen G Nash, and Ariela Sofer. *Linear and nonlinear optimization*. Siam, 2009.

[25] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[26] David Ferrucci and Adam Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

[27] Steven Bethard, Philip Ogren, and Lee Becker. ClearTK 2.0: Design patterns for machine learning in UIMA. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3289–3293, Reykjavik, Iceland, 5 2014. European Language Resources Association (ELRA).

[28] Naoaki Okazaki. CRFsuite: a fast implementation of Conditional Random Fields (CRFs), 2007.

[29] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[30] Guergana Savova, James Masanz, Philip Ogren, Jiaping Zheng, Sunghwan Sohn, Karin Kipper-Schuler, and Christopher Chute. Mayo Clinic clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. (17):507–513, 2010.

[31] L Bos et al. SNOMED-CT: The advanced terminology and coding system for eHealth. *Stud Health Technol Inform*, 121:279–290, 2006.

[32] Simon Liu, Wei Ma, Robin Moore, Vikraman Ganesan, and Stuart Nelson. RxNorm: prescription for electronic drug information exchange. *IT professional*, 7(5):17–23, 2005.

[33] Showick Thorpe. Importance of English language in higher education. `http://www.indiaeducationreview.com/article/importance-english-language-higher-education/17295`, February 2015.

[34] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009.

[35] Unicode Consortium et al. The Unicode standard, version 6.0. 0,(mountain view, ca: The unicode consortium, 2011. isbn 978-1-936213-01-6), 2010.

[36] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *arXiv preprint arXiv:1106.1813*, 2011.

[37] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 2004: PROCEEDINGS OF THE TWENTY-FIRST INTERNATIONAL CONFERENCE ON MACHINE LEARNING. OMNIPRESS*, pages 919–926, 2004.

[38] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128, 2006.

A Common groups in Regular Expressions

Table 17: Commonly used predefined groups of regular expressions.

RegEx syntax	**ASCII chars**	**Description**
`\w`	`[A-Za-z0-9_]`	Alphanumeric characters plus "_"
`\W`	`[^A-Za-z0-9_]`	Non-word characters
`\d`	`[0-9]`	Digits
`\D`	`[^0-9]`	Non-digits
`\s`	`[ \t\r\n\v\f]`	Whitespace character
`\S`	`[^ \t\r\n\v\f]`	Non-whitespace character
`\b`	`(?<=\W)(?=\w)|(?<=\w)(?=\W)`	Word boundary

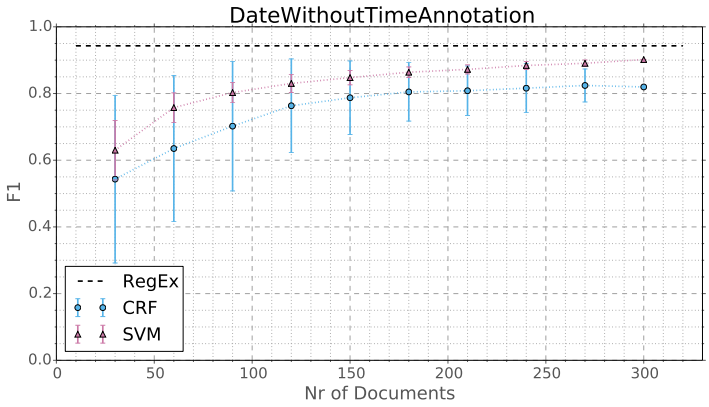
B Unicode Character Type

Table 18: Character types according to the Unicode General Category

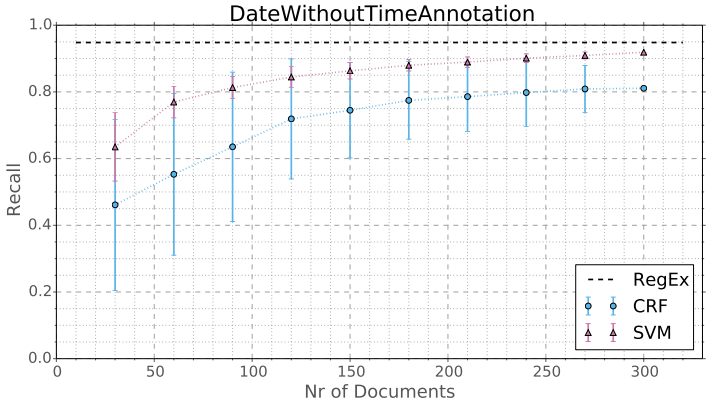
Short Code	Description
Lu	Letter, uppercase
Ll	Letter, lowercase
Lt	Letter, titlecase
Lm	Letter, modifier
Lo	Letter, other
Mn	Mark, nonspacing
Mc	Mark, spacing combining
Me	Mark, enclosing
Nd	Number, decimal digit
Nl	Number, letter
No	Number, other
Pc	Punctuation, connector
Pd	Punctuation, dash
Ps	Punctuation, open
Pe	Punctuation, close
Pi	Punctuation, initial quote (may behave like Ps or Pe depending on usage)
Pf	Punctuation, final quote (may behave like Ps or Pe depending on usage)
Po	Punctuation, other
Sm	Symbol, math
Sc	Symbol, currency
Sk	Symbol, modifier
So	Symbol, other
Zs	Separator, space
Zl	Separator, line
Zp	Separator, paragraph
Cc	Other, control
Cf	Other, format
Cs	Other, surrogate
Co	Other, private use
Cn	Other, not assigned (including noncharacters)

C Varying Number of Documents

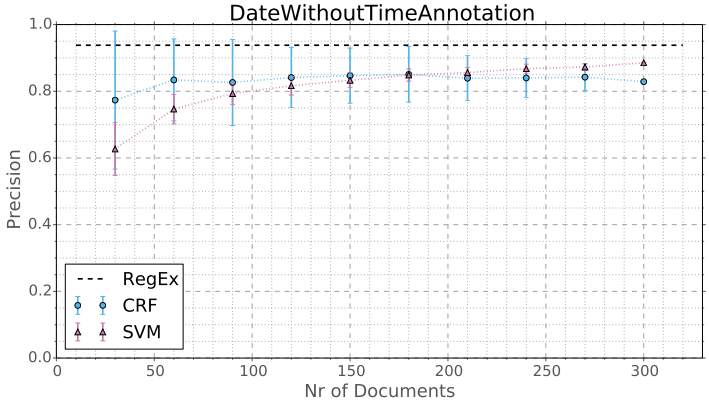
C.1 Date



(a) Date F_1-score, covering criteria.



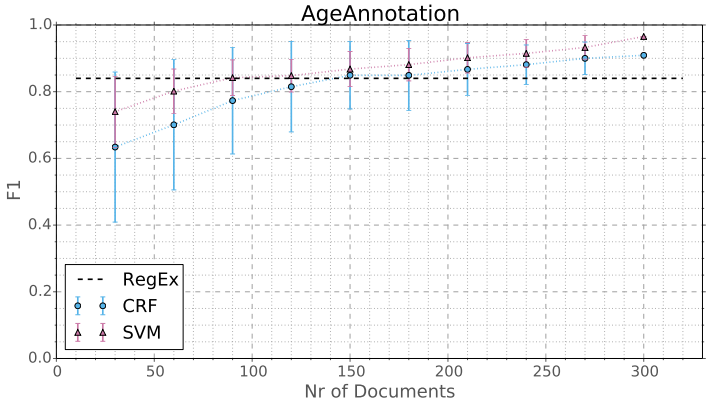
(b) Date recall, covering criteria.



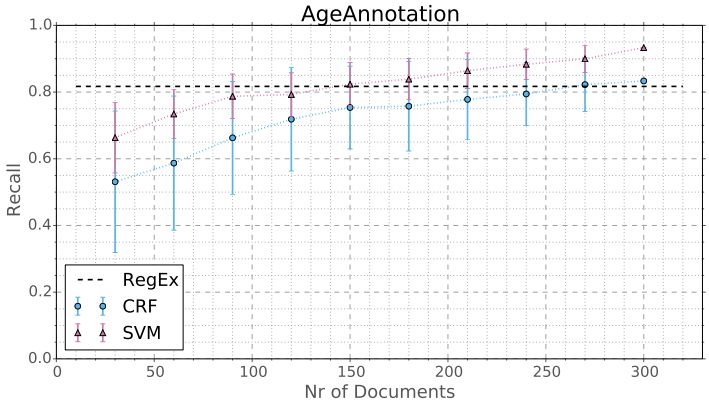
(c) Date precision, covering criteria.

Figure 21: Results when varying amount of available records

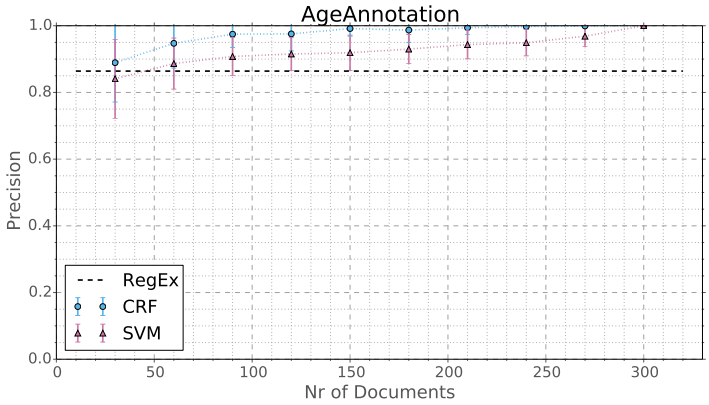
C.2 Age



(a) Age F_1-score, covering criteria.



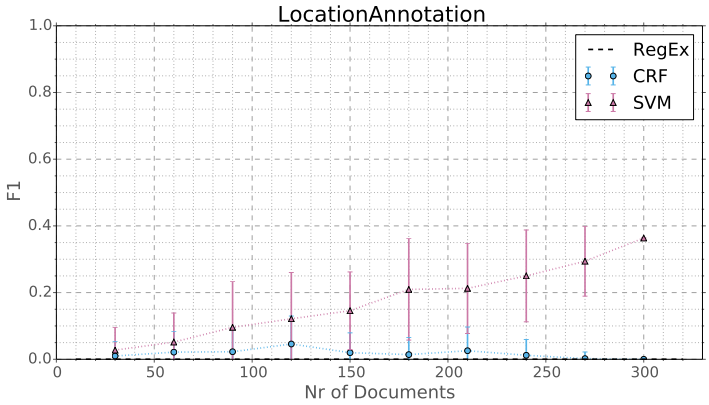
(b) Age recall, covering criteria.



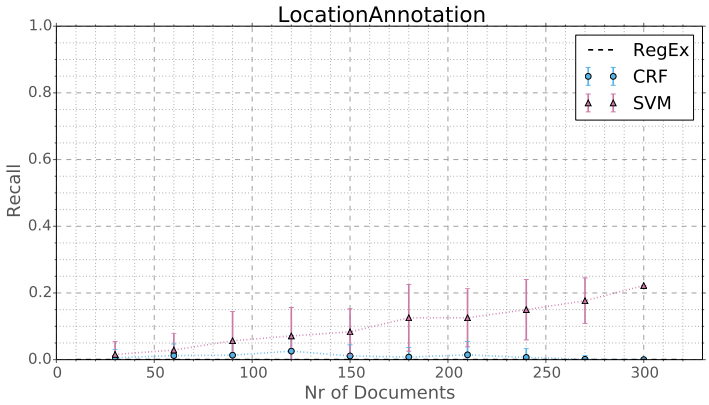
(c) Age precision, covering criteria.

Figure 22: Results when varying amount of available records

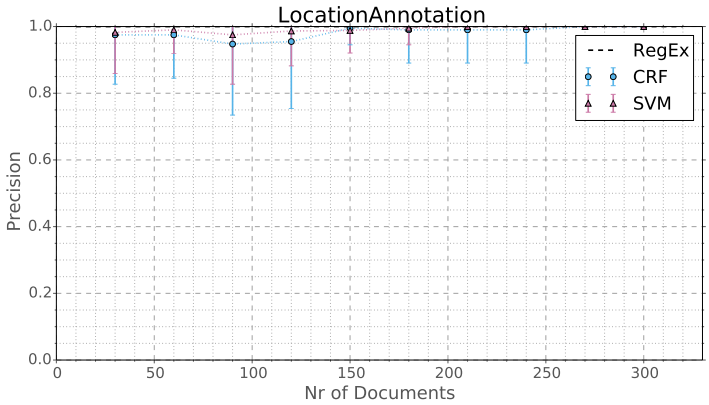
C.3 Location



(a) Location F_1-score, covering criteria.



(b) Location recall, covering criteria.



(c) Location precision, covering criteria.

Figure 23: Results when varying amount of available records

D Feature Ranking

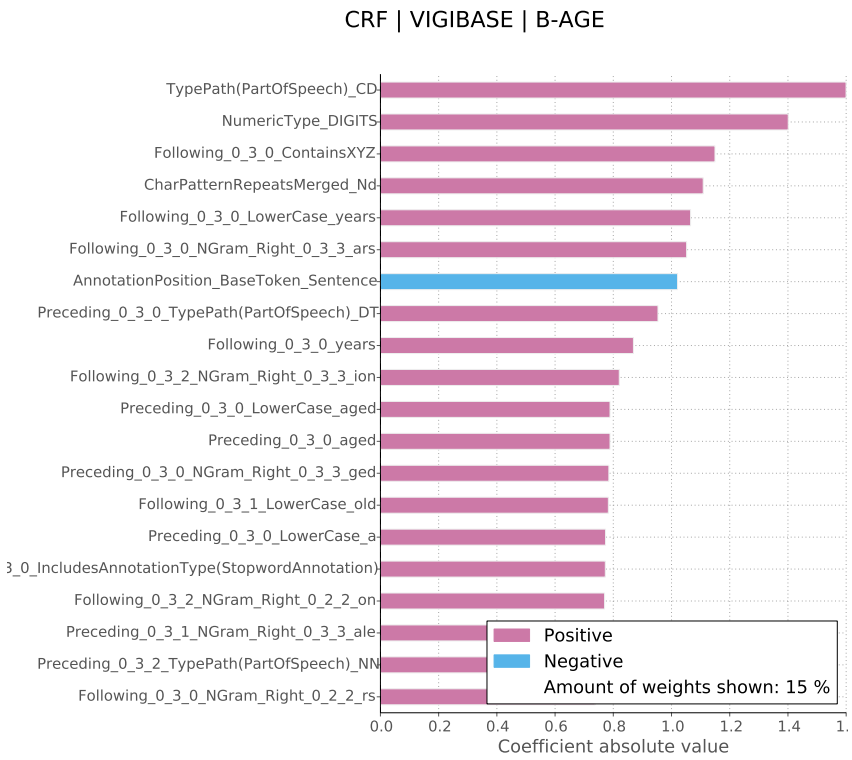


Figure 24: Top 20 features using CRF on the VigiBase data set for category B-Age

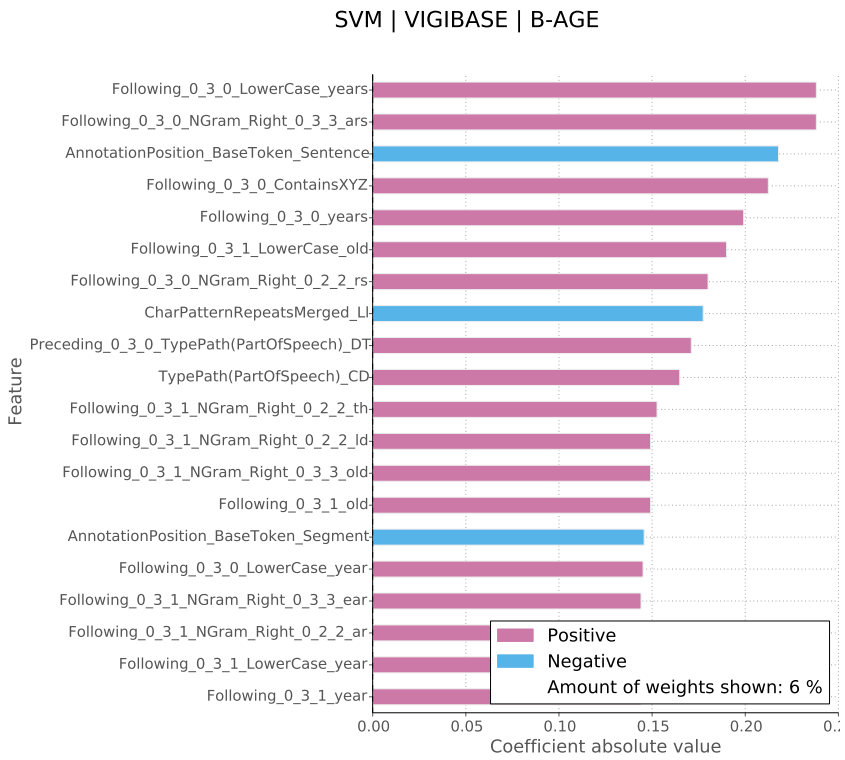


Figure 25: Top 20 features using SVM on the VigiBase data set for category B-Age

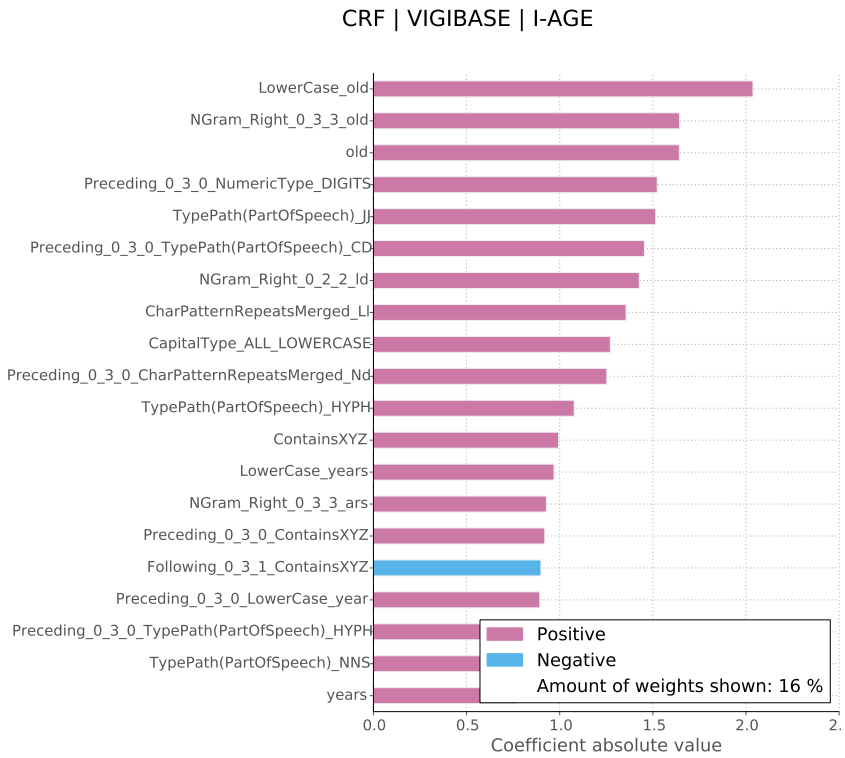


Figure 26: Top 20 features using CRF on the VigiBase data set for category I-Age

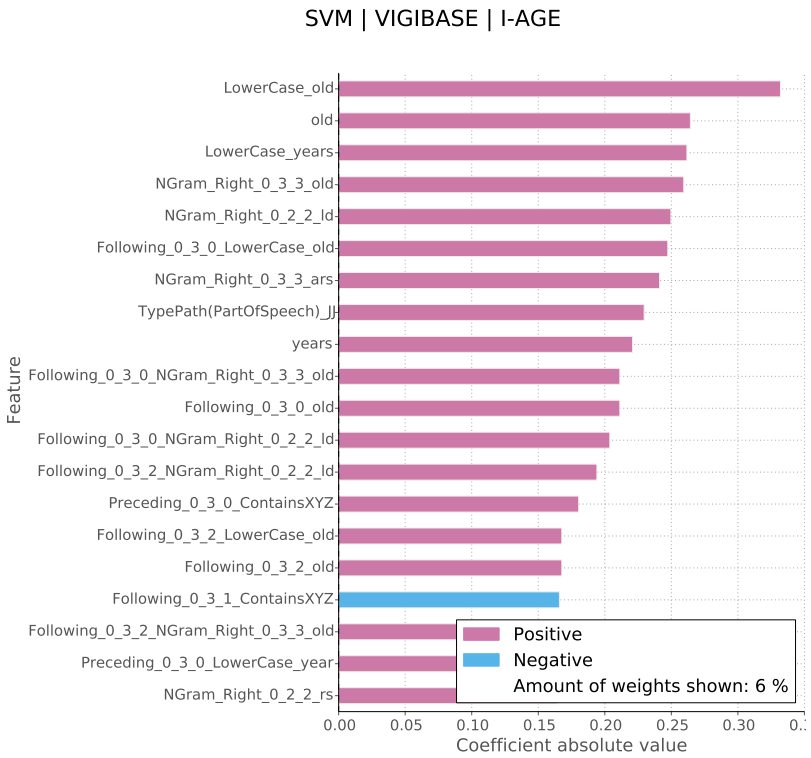


Figure 27: Top 20 features using SVM on the VigiBase data set for category I-Age

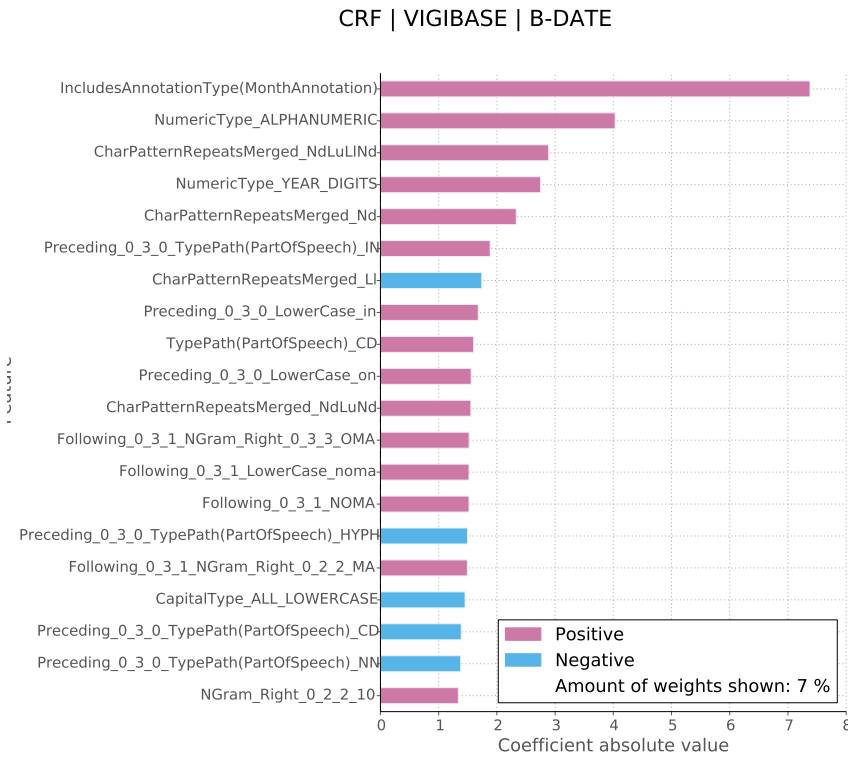


Figure 28: Top 20 features using CRF on the VigiBase data set for category B-Date

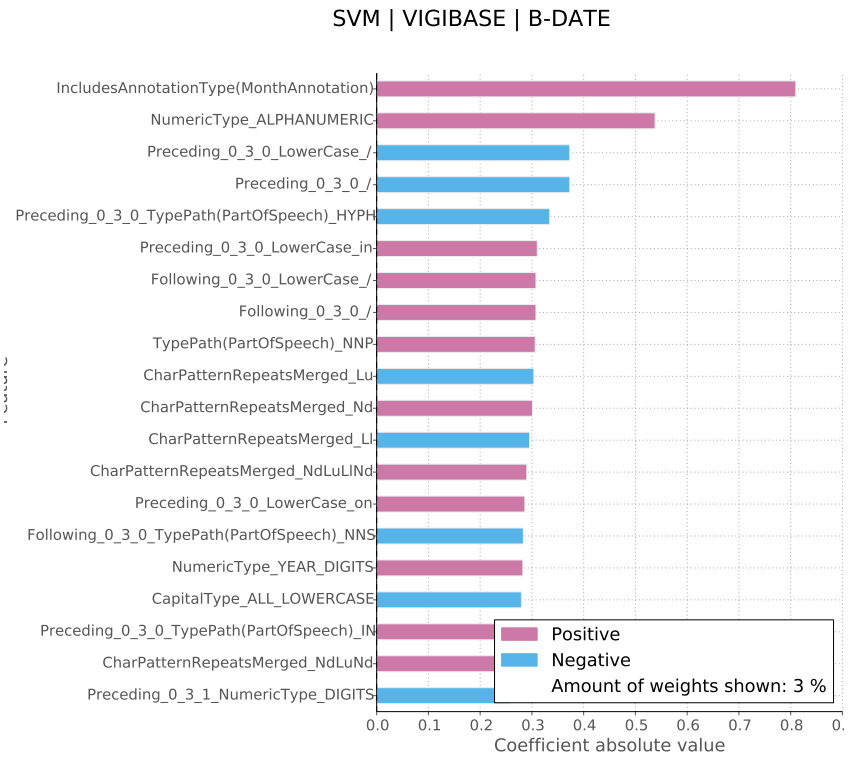


Figure 29: Top 20 features using SVM on the VigiBase data set for category B-Date

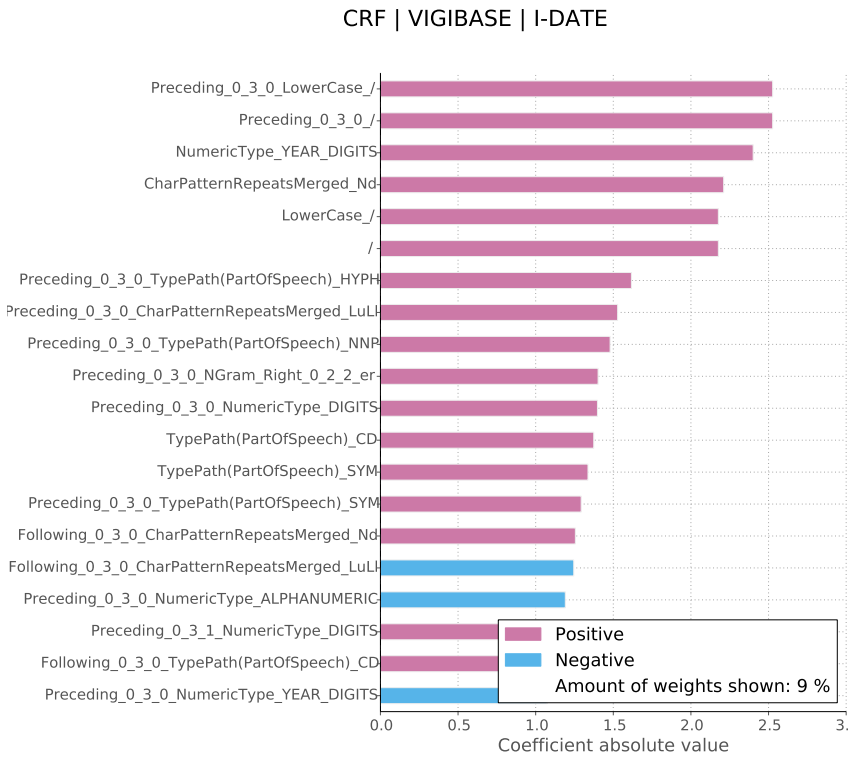


Figure 30: Top 20 features using CRF on the VigiBase data set for category I-Date

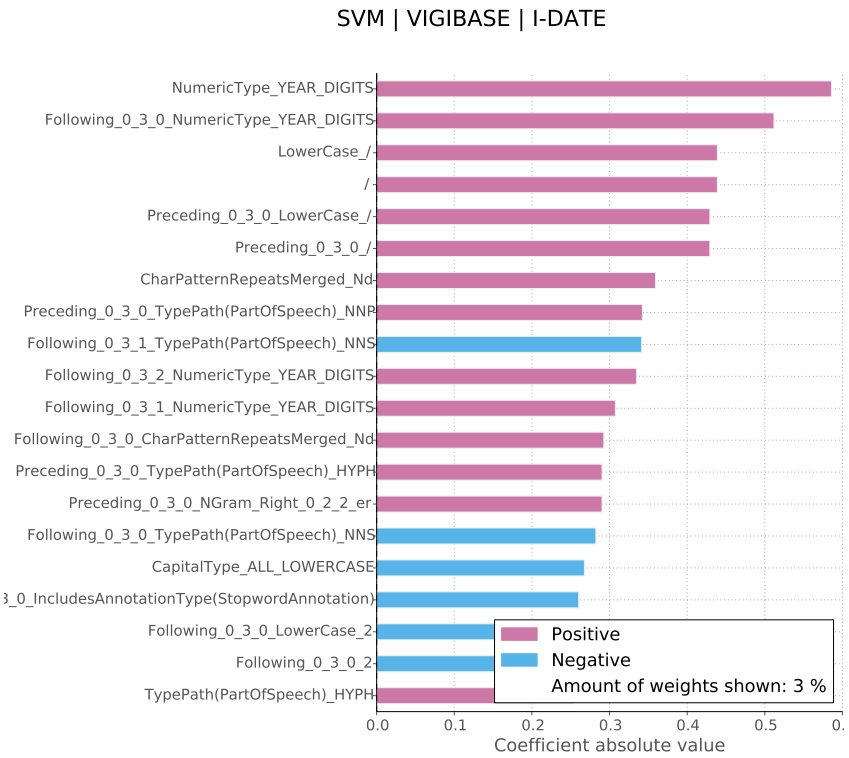


Figure 31: Top 20 features using SVM on the VigiBase data set for category I-Date

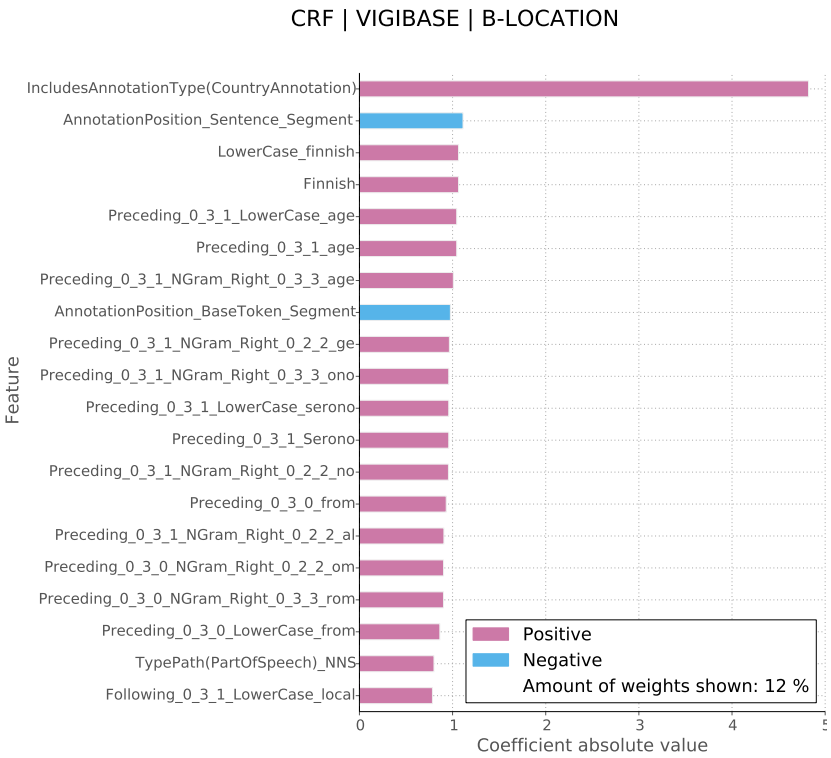


Figure 32: Top 20 features using CRF on the VigiBase data set for category B-Location

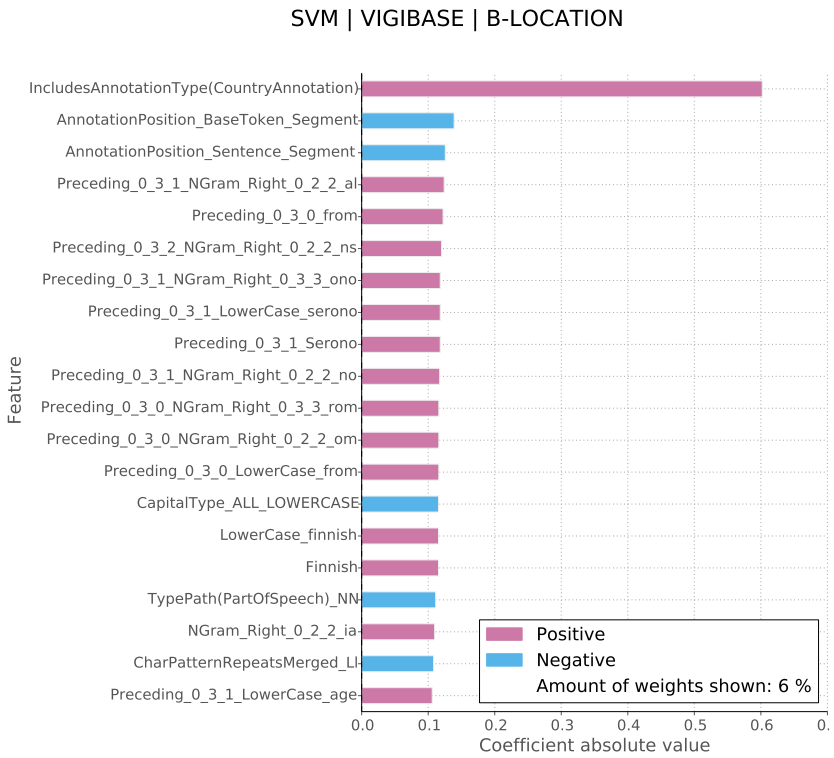


Figure 33: Top 20 features using SVM on the VigiBase data set for category B-Location

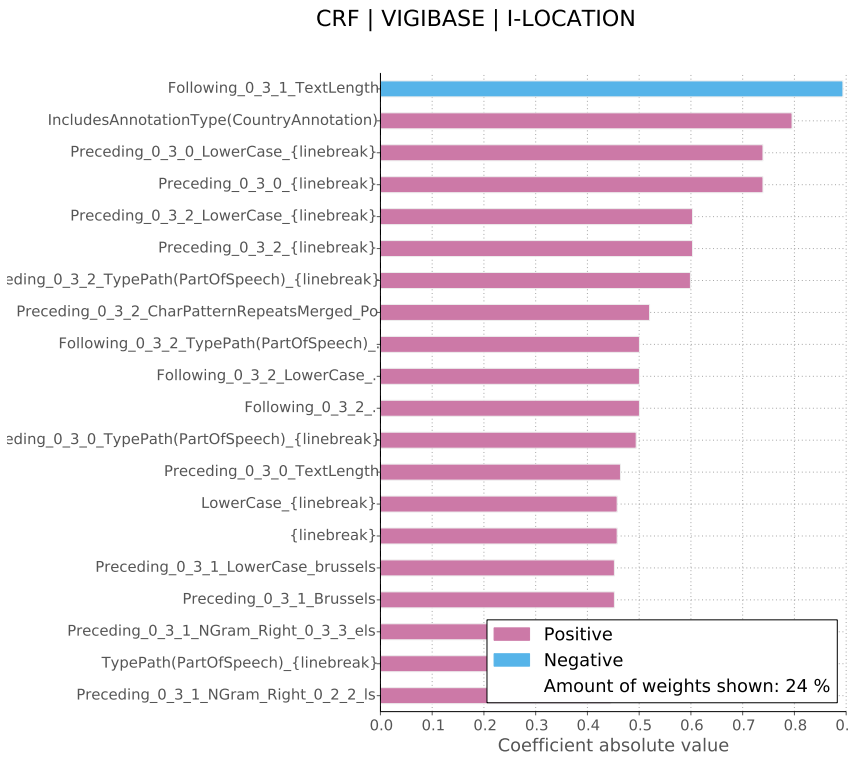


Figure 34: Top 20 features using CRF on the VigiBase data set for category I-Location

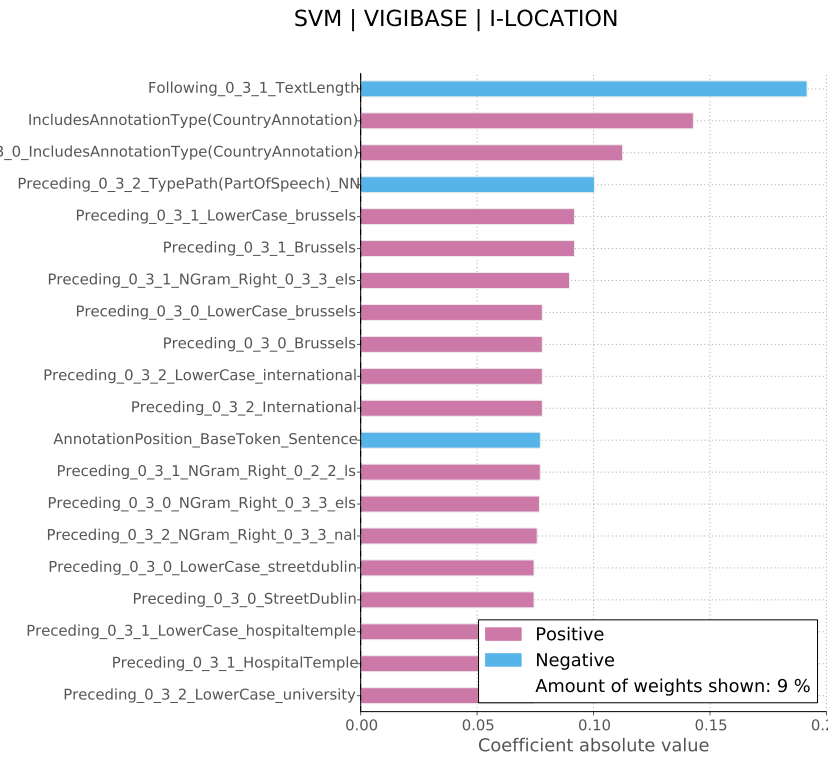


Figure 35: Top 20 features using SVM on the VigiBase data set for category I-Location

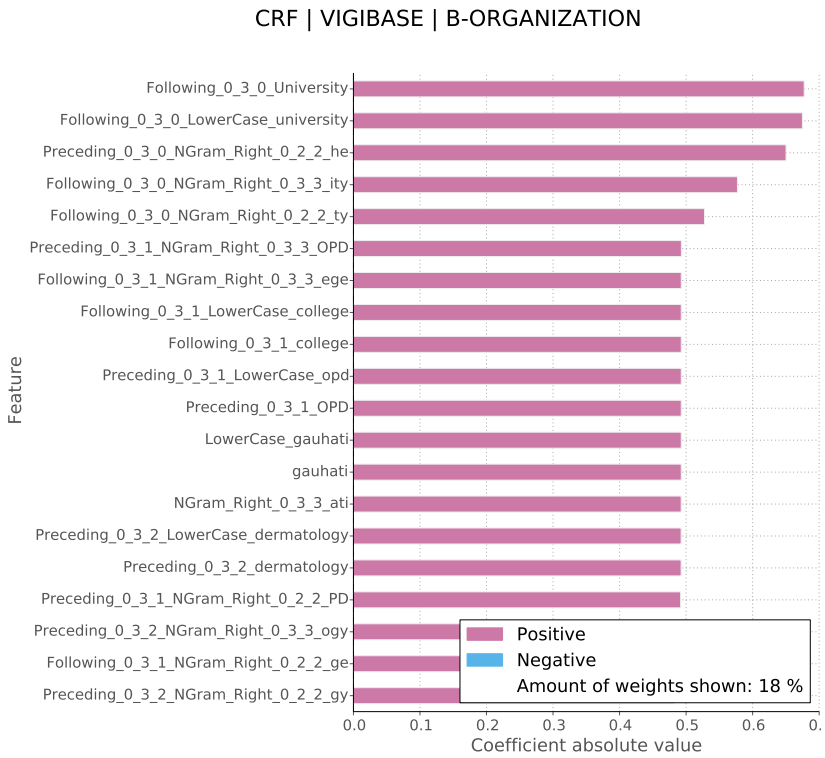


Figure 36: Top 20 features using CRF on the VigiBase data set for category B-Organization

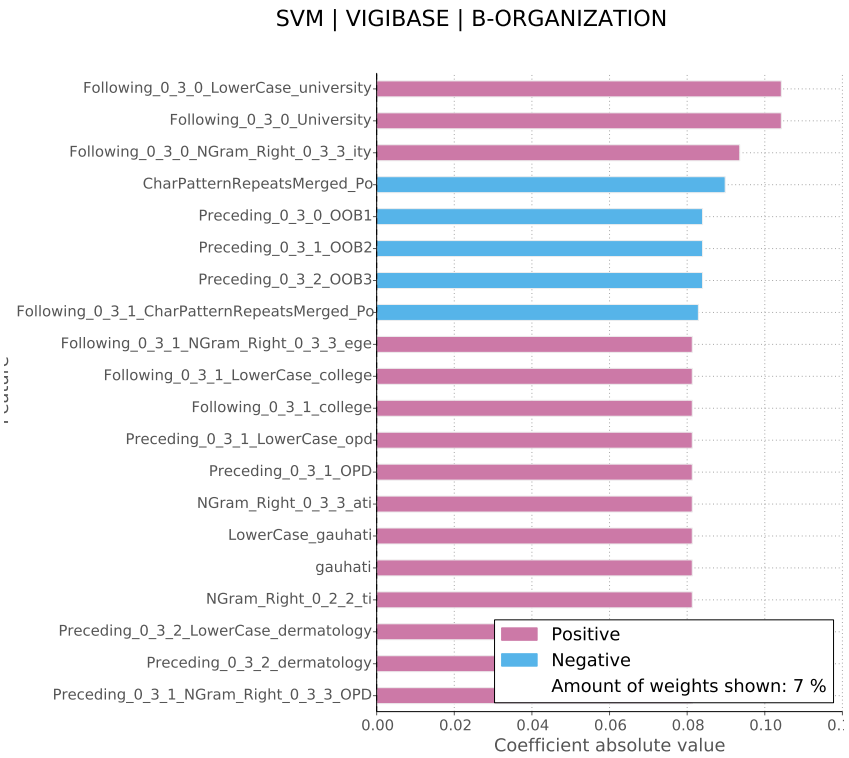


Figure 37: Top 20 features using SVM on the VigiBase data set for category B-Organization

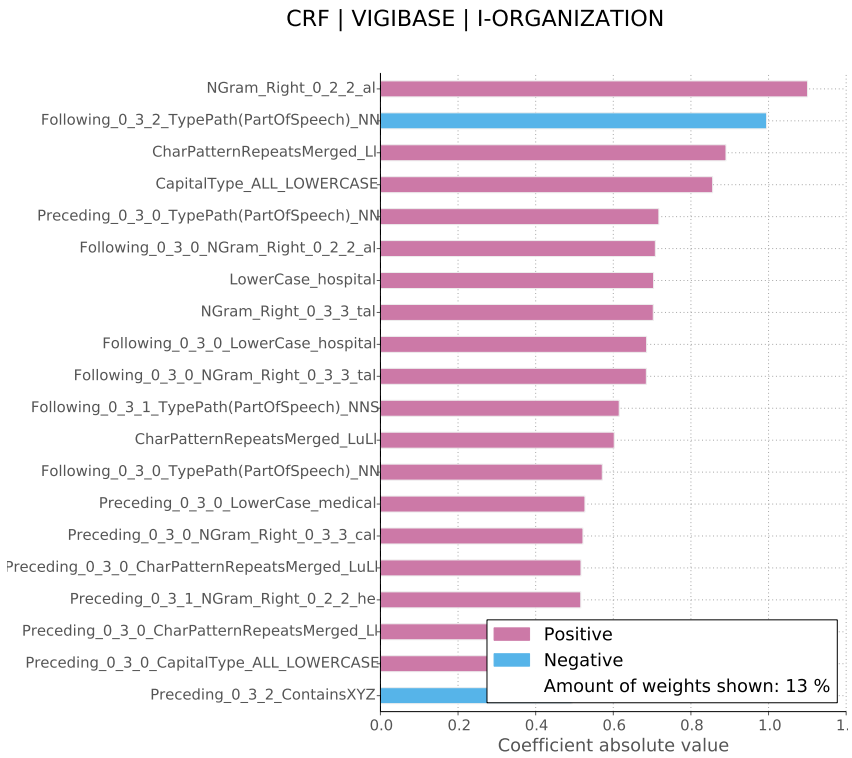


Figure 38: Top 20 features using CRF on the VigiBase data set for category I-Organization

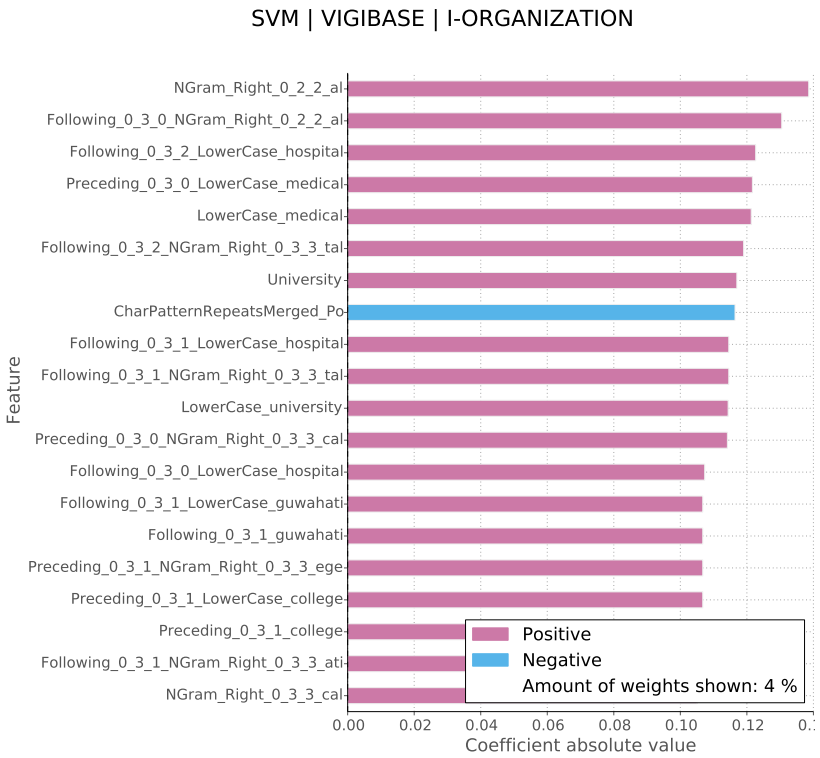


Figure 39: Top 20 features using SVM on the VigiBase data set for category I-Organization

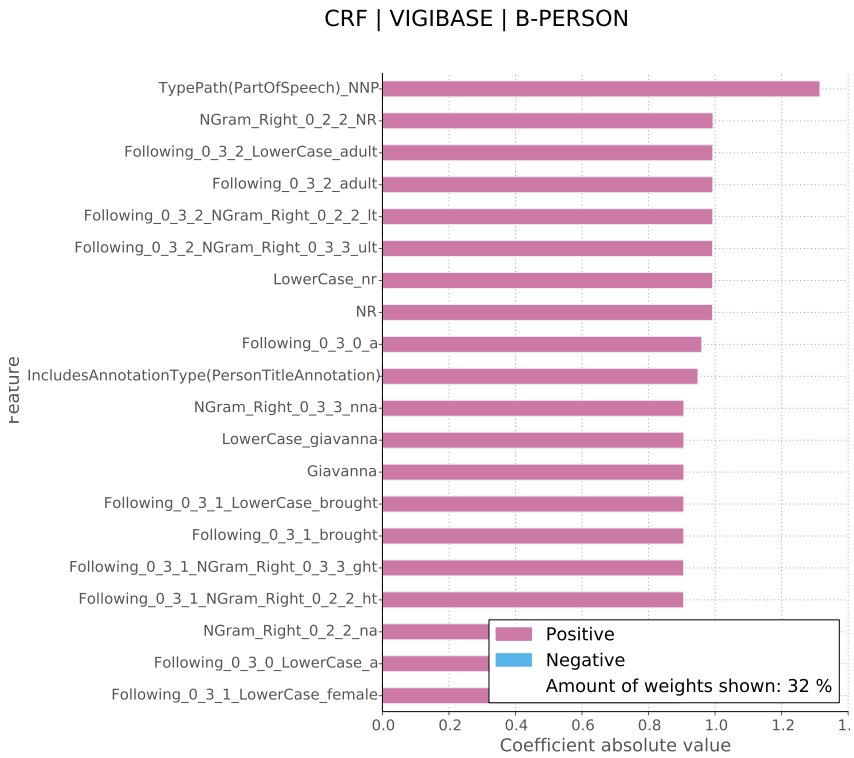


Figure 40: Top 20 features using CRF on the VigiBase data set for category B-Person

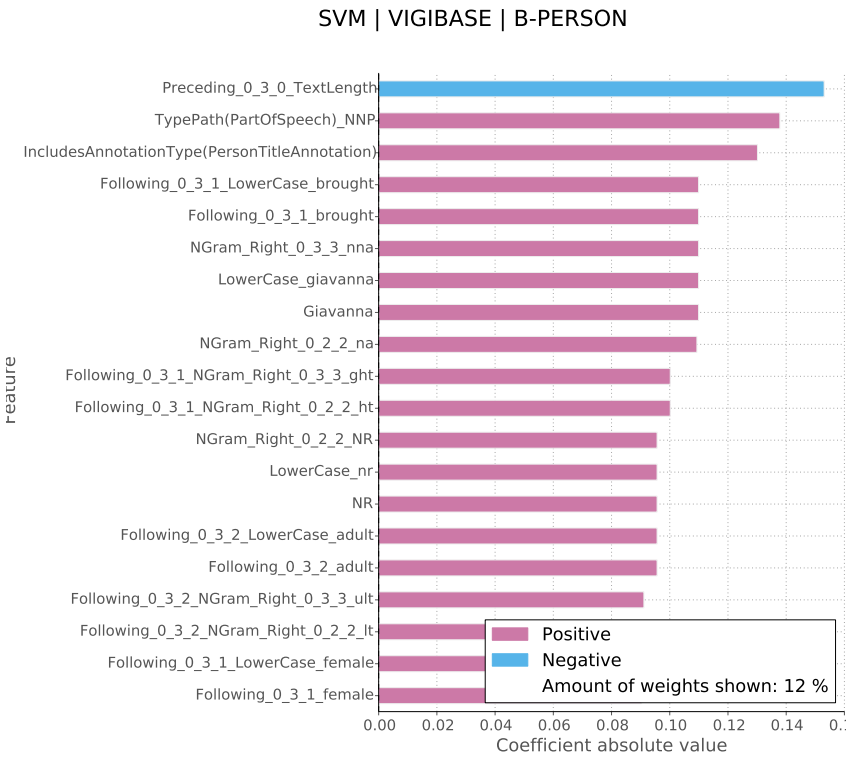


Figure 41: Top 20 features using SVM on the VigiBase data set for category B-Person

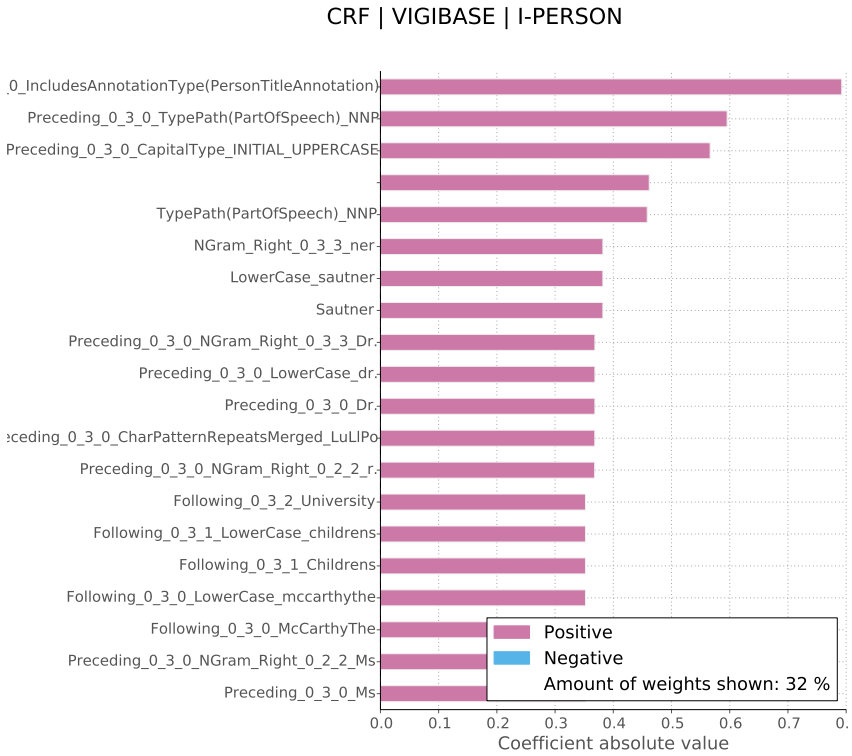


Figure 42: Top 20 features using CRF on the VigiBase data set for category I-Person

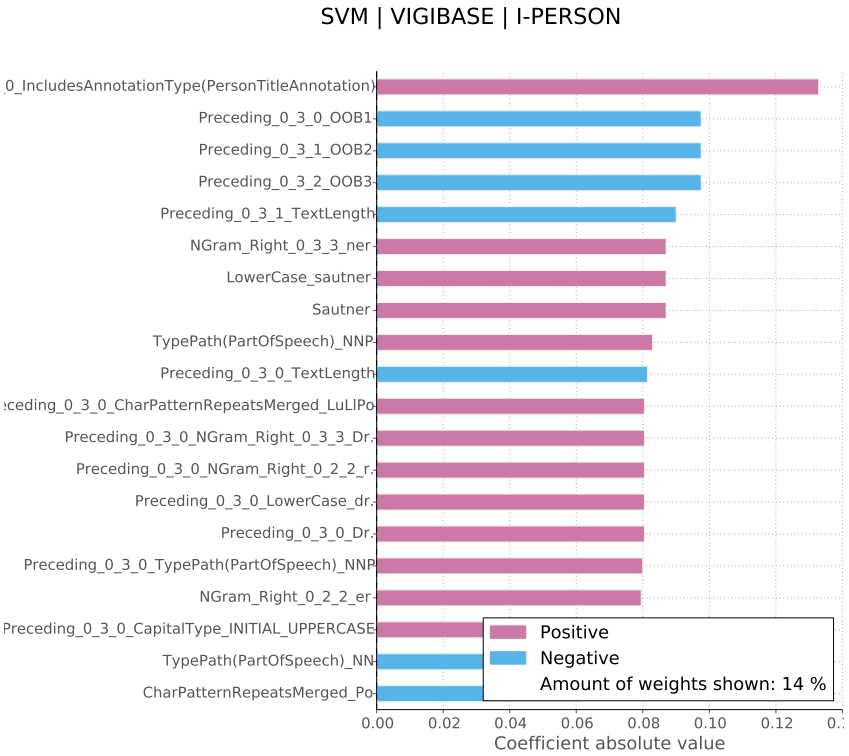


Figure 43: Top 20 features using SVM on the VigiBase data set for category I-Person

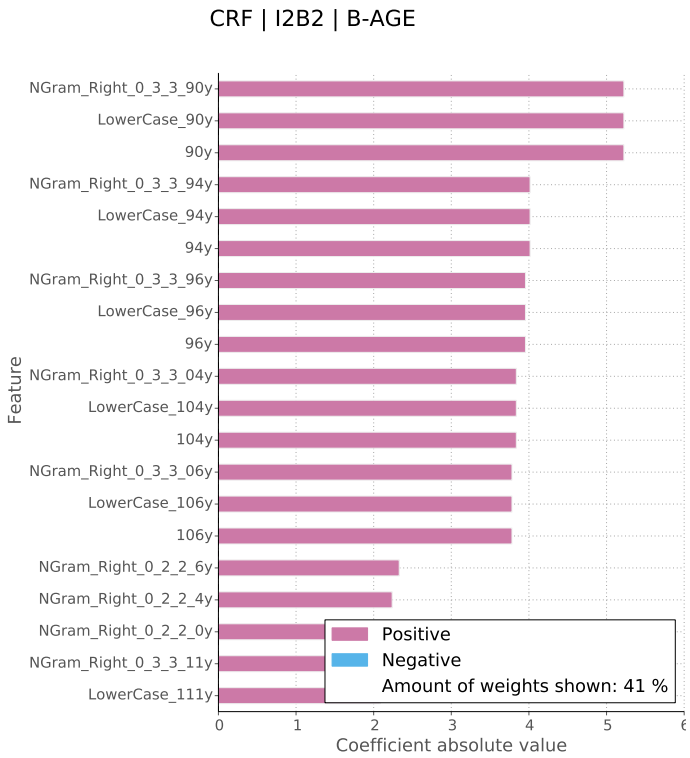


Figure 44: Top 20 features using CRF on the i2b2 data set for category B-Age

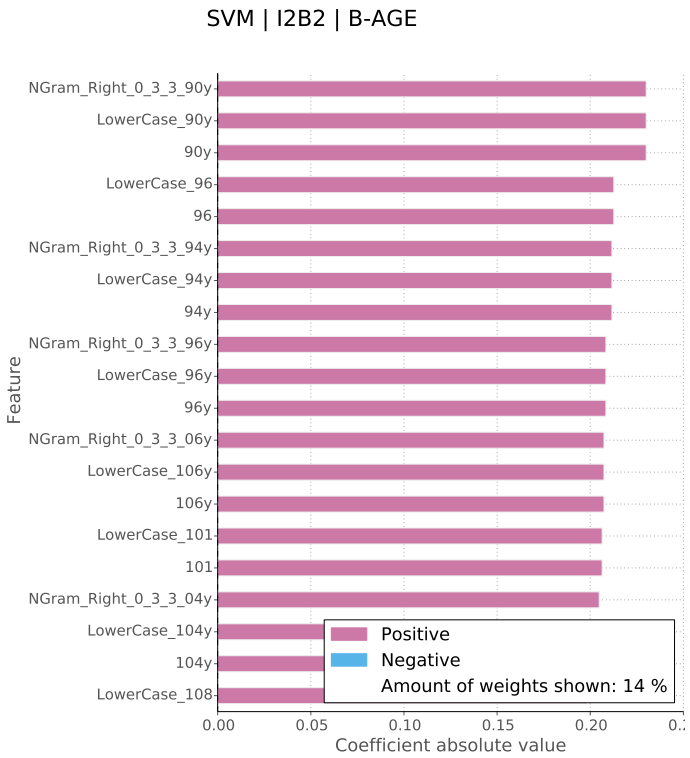


Figure 45: Top 20 features using SVM on the i2b2 data set for category B-Age

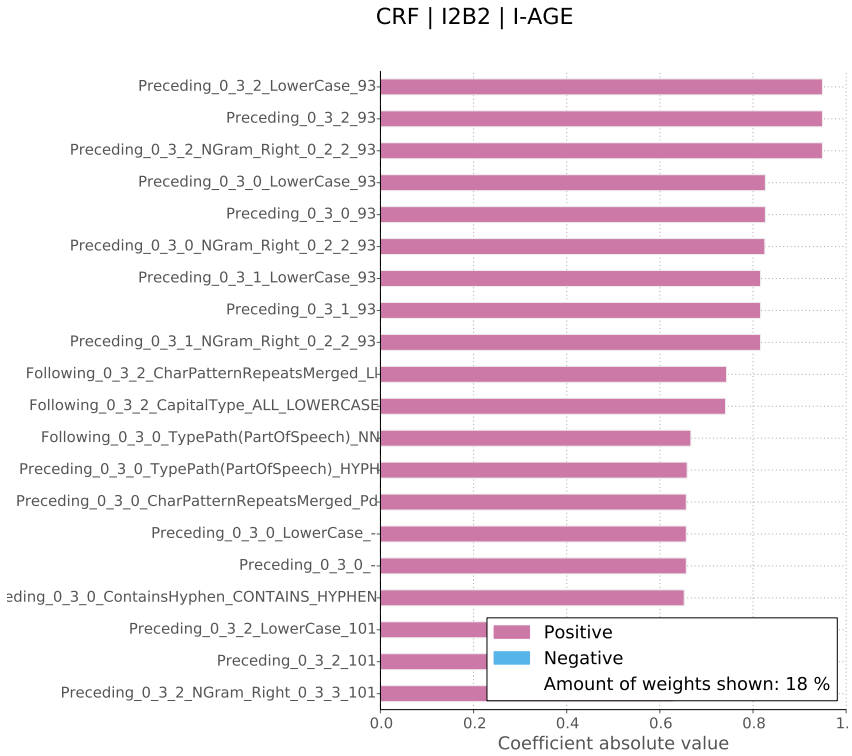


Figure 46: Top 20 features using CRF on the i2b2 data set for category I-Age

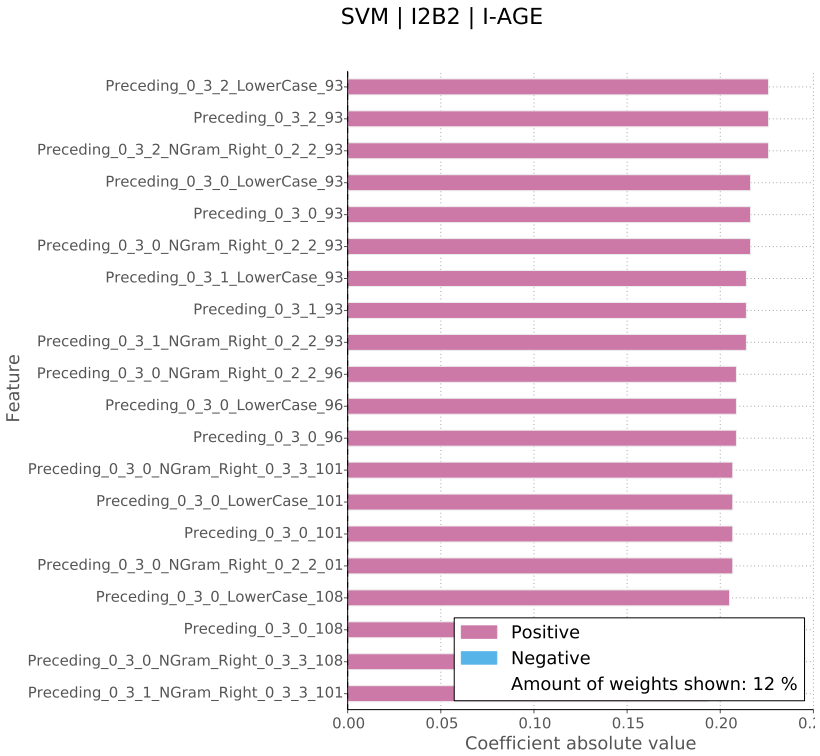


Figure 47: Top 20 features using SVM on the i2b2 data set for category I-Age

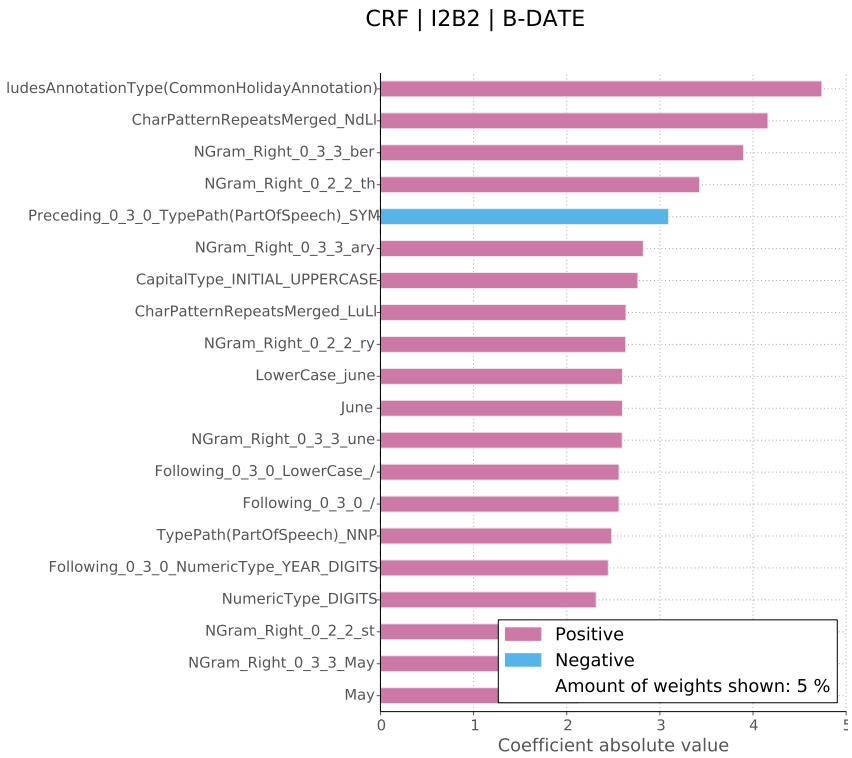


Figure 48: Top 20 features using CRF on the i2b2 data set for category B-Date

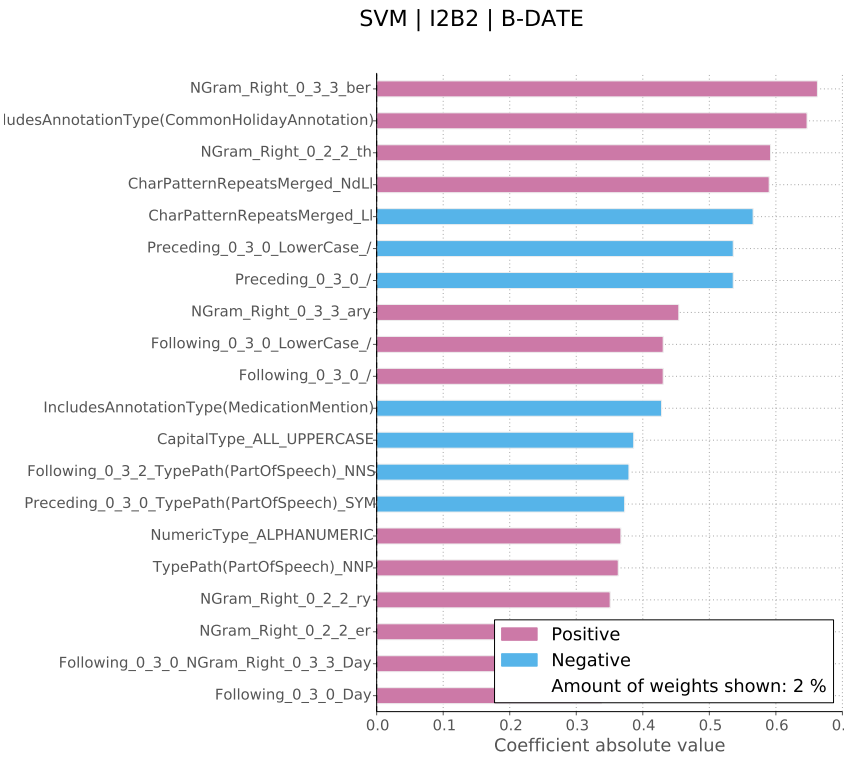


Figure 49: Top 20 features using SVM on the i2b2 data set for category B-Date

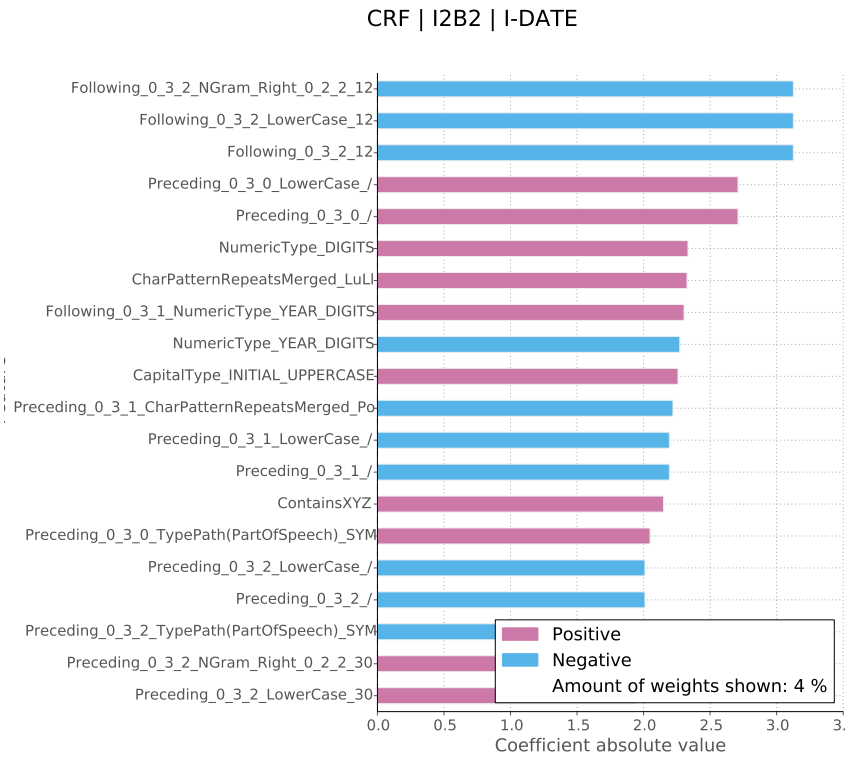


Figure 50: Top 20 features using CRF on the i2b2 data set for category I-Date

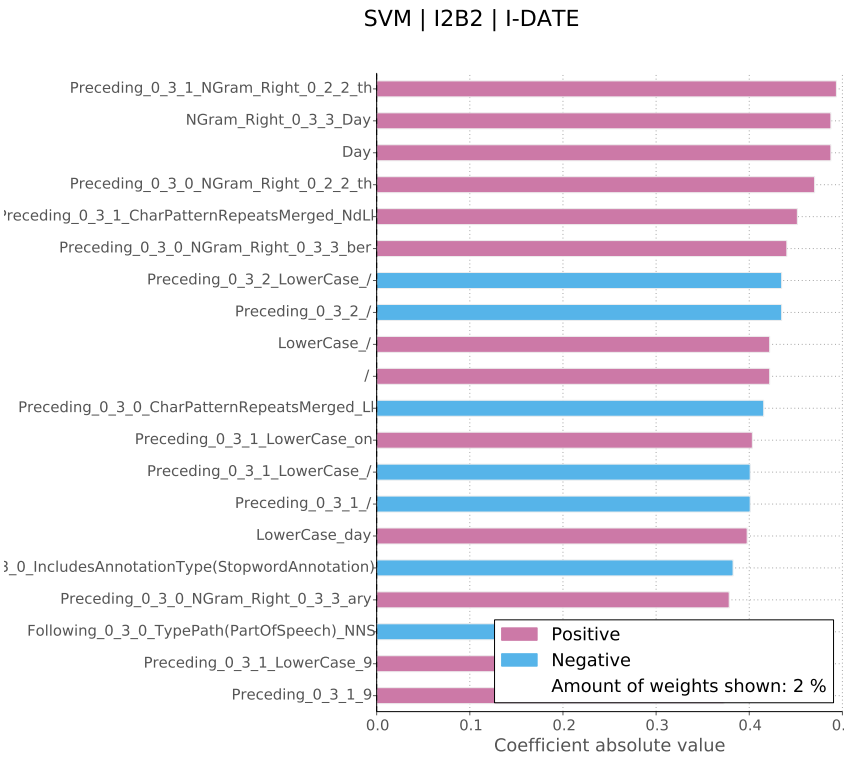


Figure 51: Top 20 features using SVM on the i2b2 data set for category I-Date

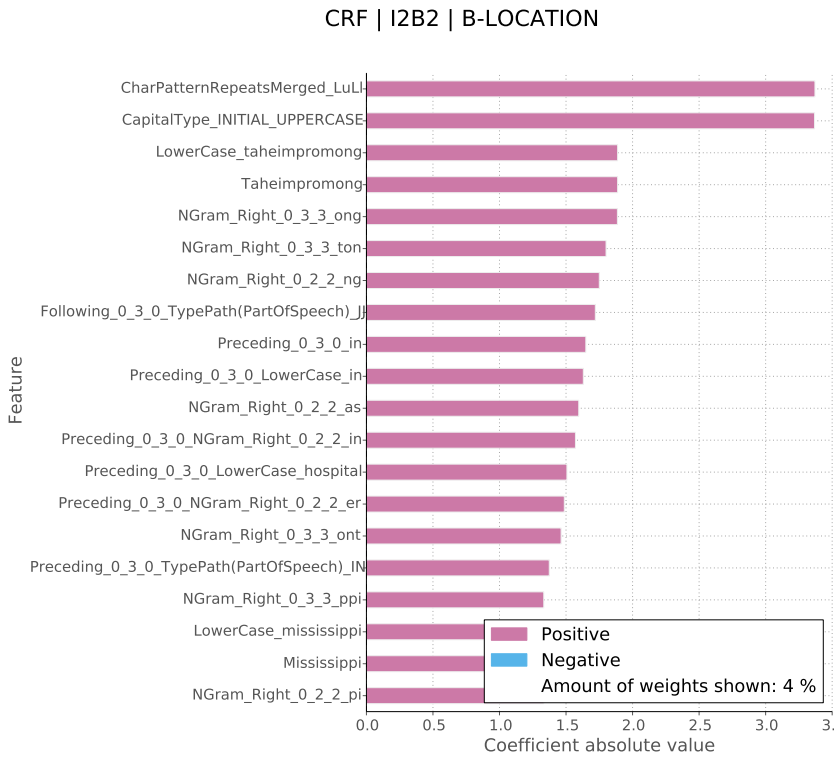


Figure 52: Top 20 features using CRF on the i2b2 data set for category B-Location

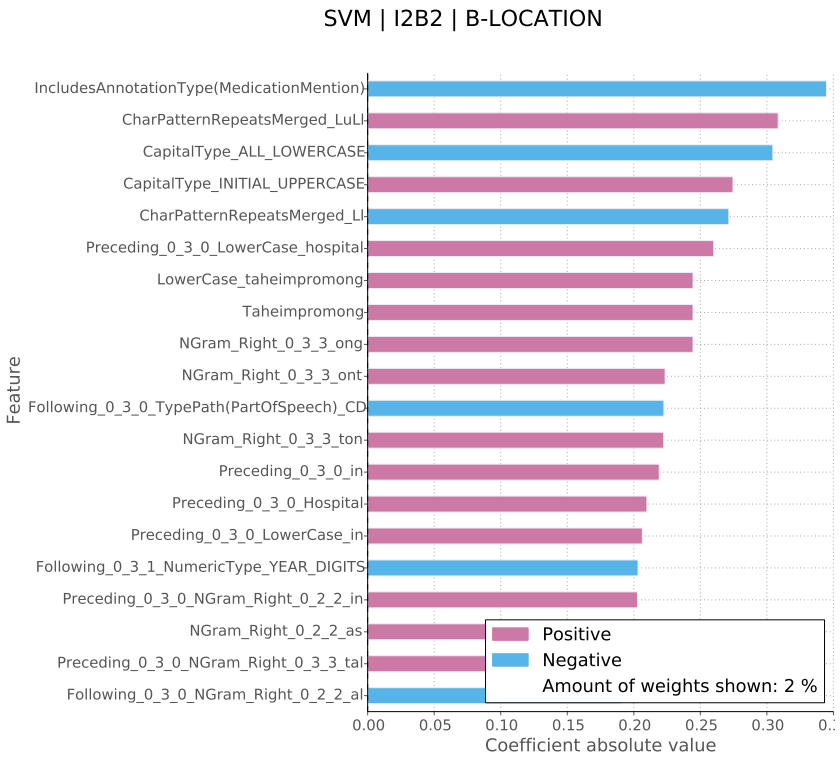


Figure 53: Top 20 features using SVM on the i2b2 data set for category B-Location

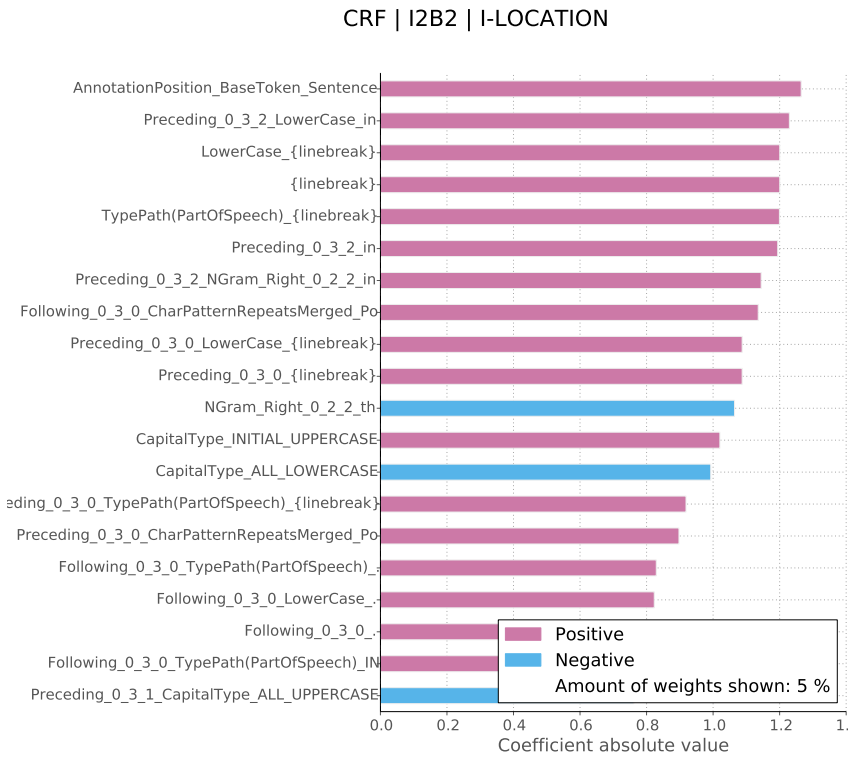


Figure 54: Top 20 features using CRF on the i2b2 data set for category I-Location

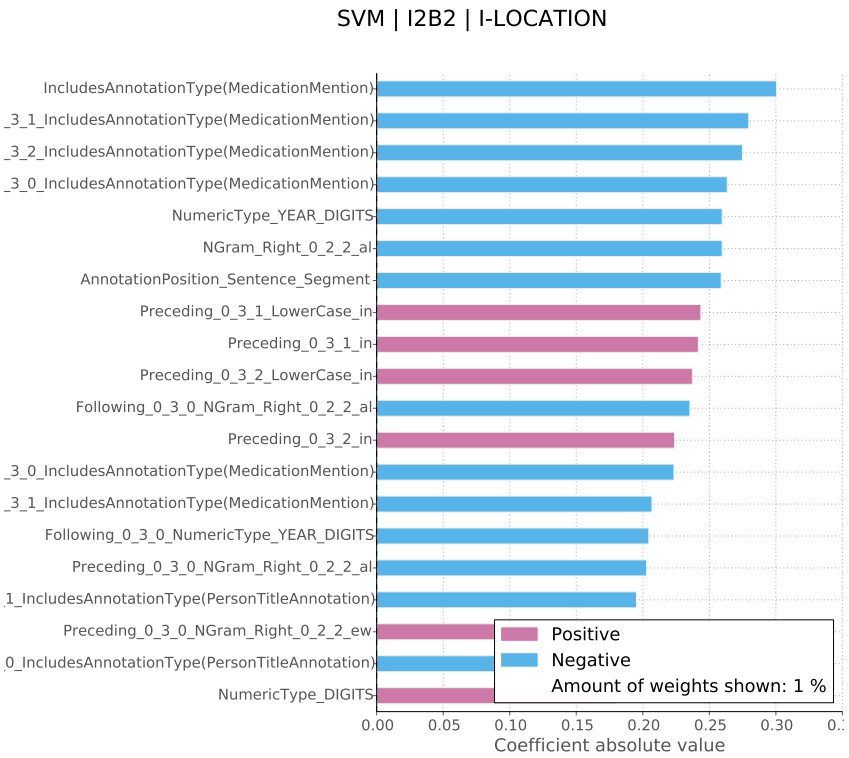


Figure 55: Top 20 features using SVM on the i2b2 data set for category I-Location

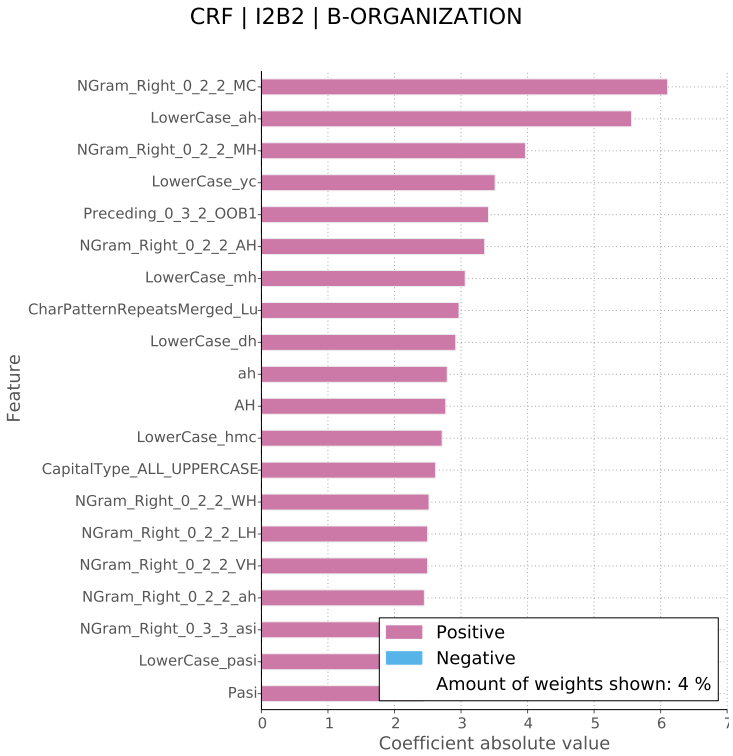


Figure 56: Top 20 features using CRF on the i2b2 data set for category B-Organization

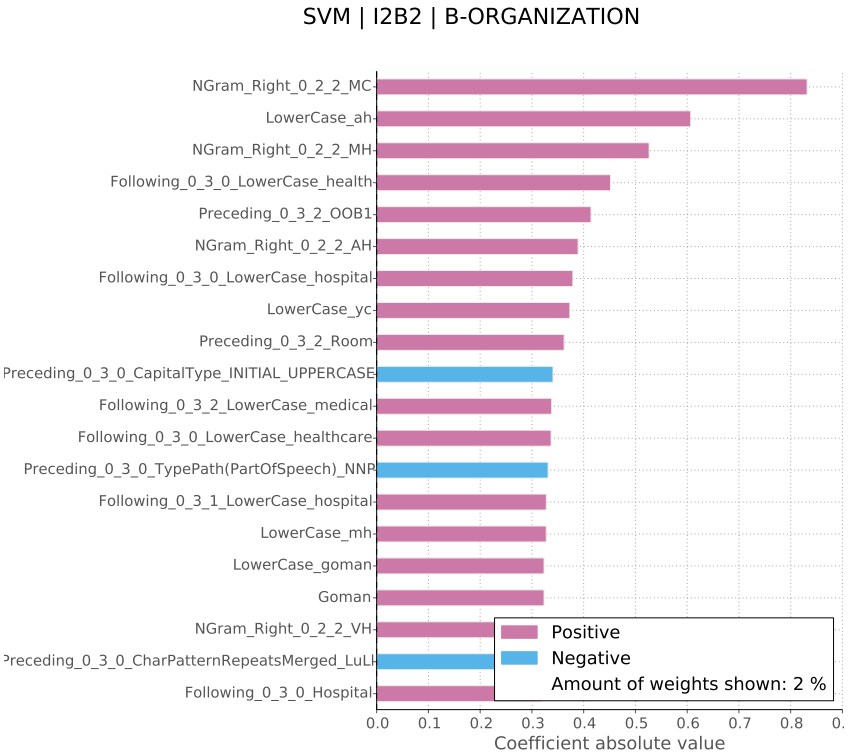


Figure 57: Top 20 features using SVM on the i2b2 data set for category B-Organization

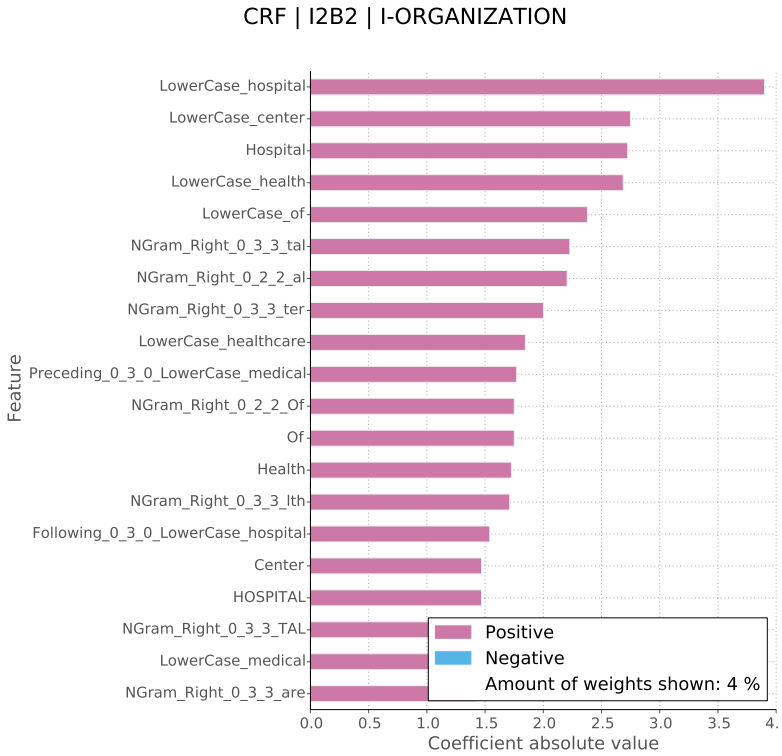


Figure 58: Top 20 features using CRF on the i2b2 data set for category I-Organization

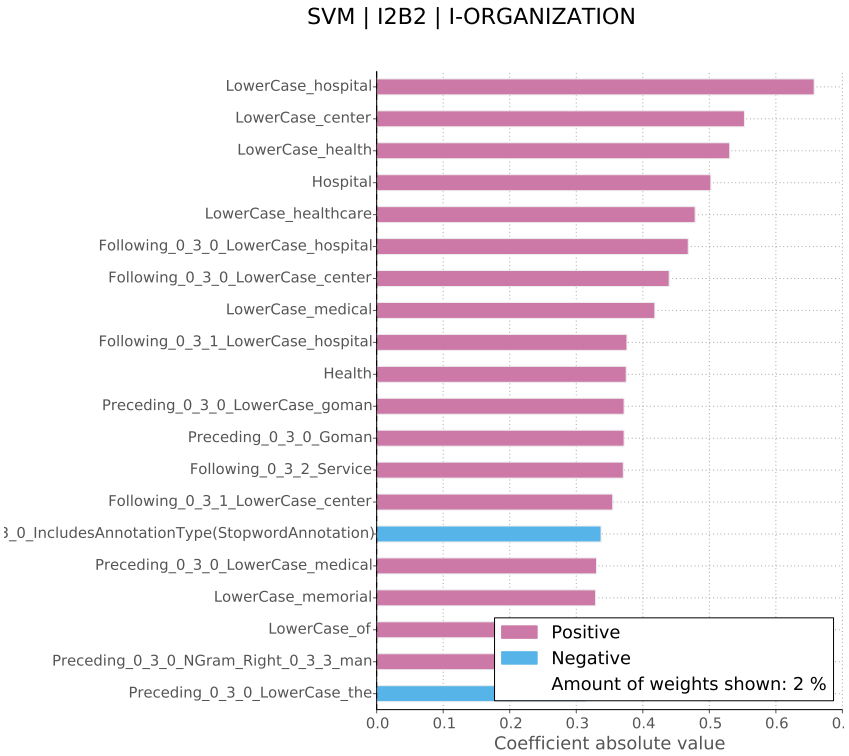


Figure 59: Top 20 features using SVM on the i2b2 data set for category I-Organization

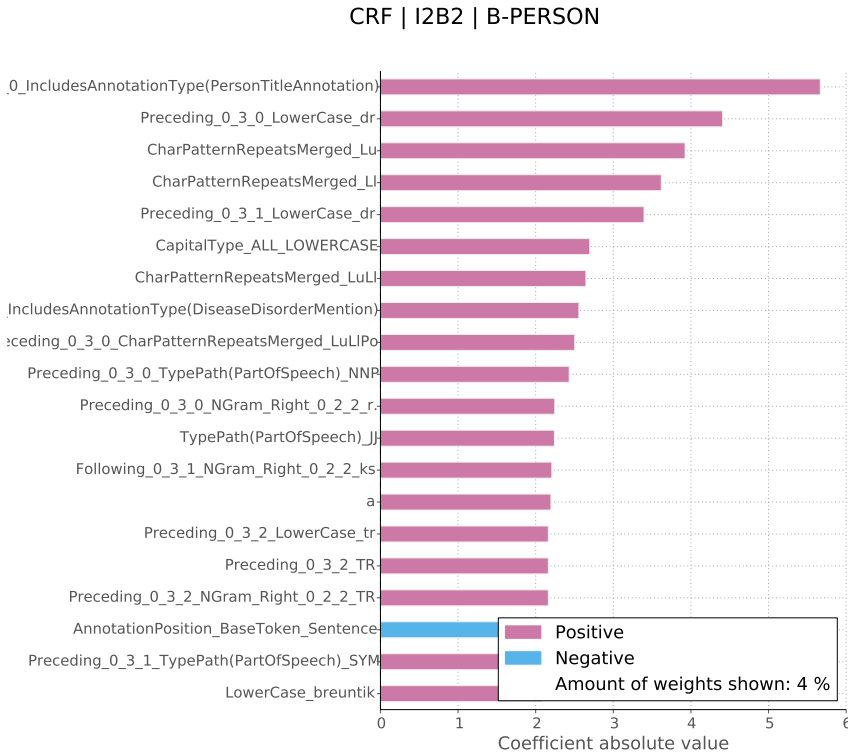


Figure 60: Top 20 features using CRF on the i2b2 data set for category B-Person

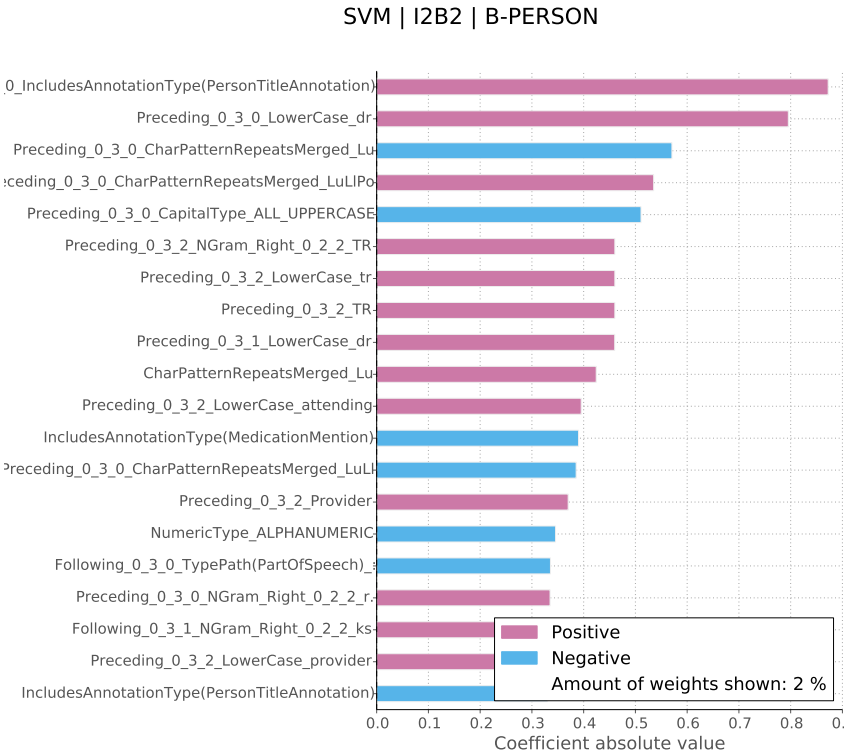


Figure 61: Top 20 features using SVM on the i2b2 data set for category B-Person

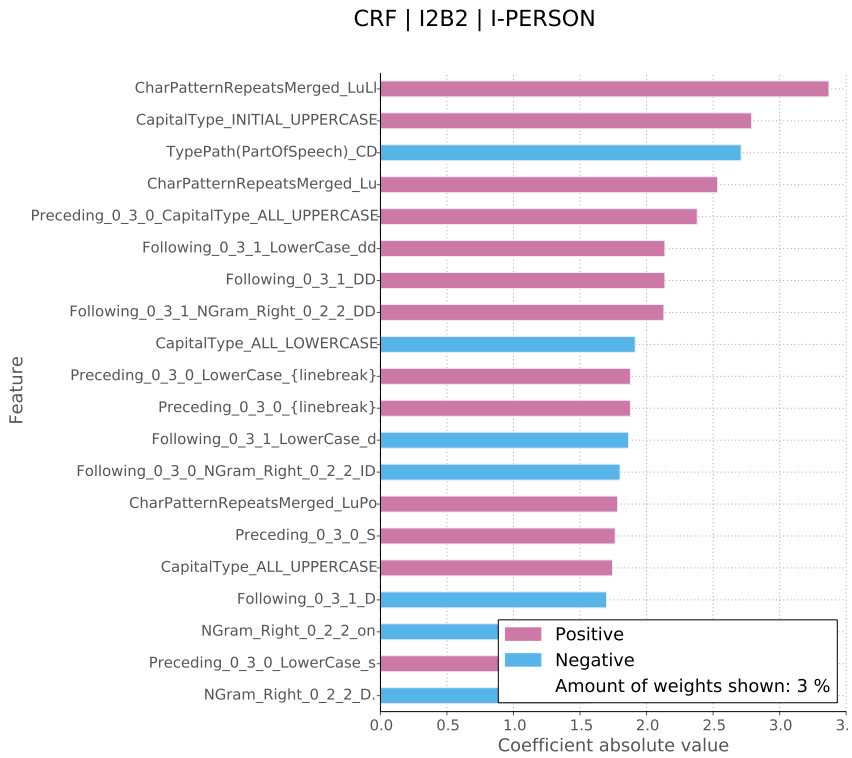


Figure 62: Top 20 features using CRF on the i2b2 data set for category I-Person

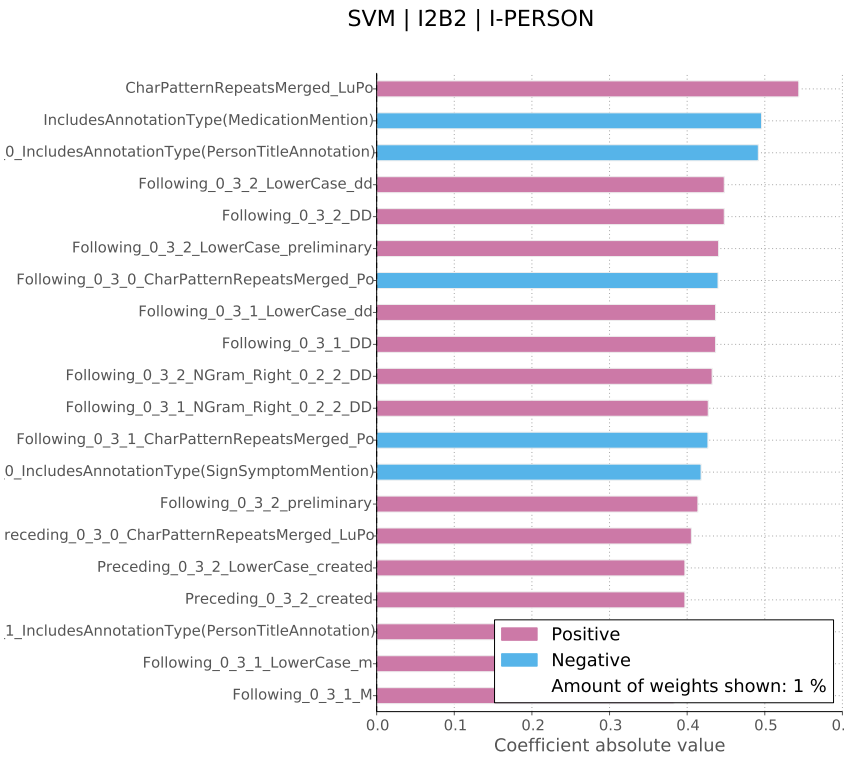


Figure 63: Top 20 features using SVM on the i2b2 data set for category I-Person

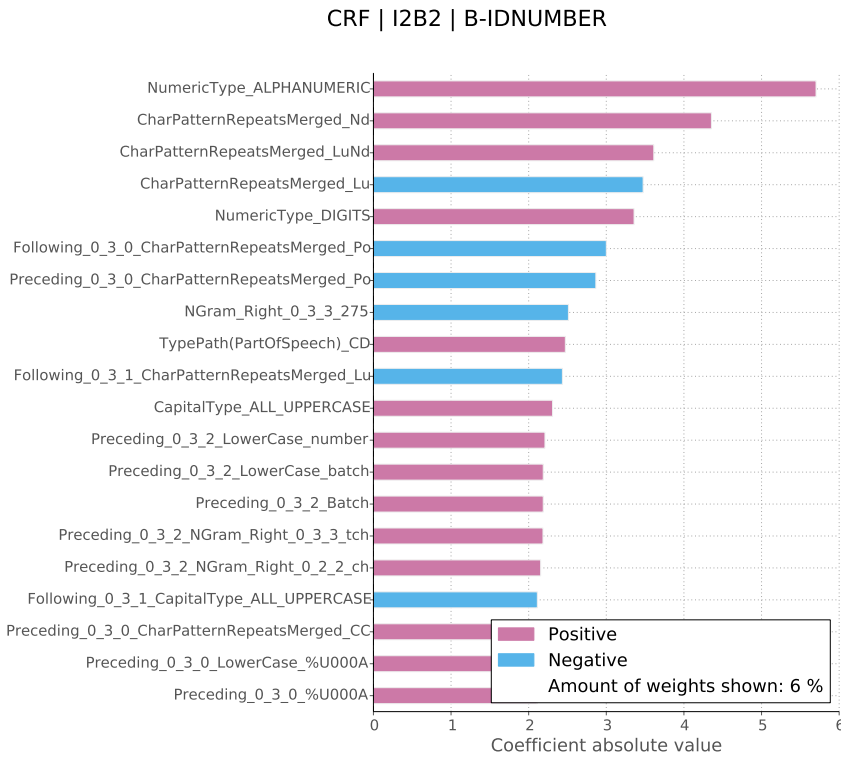


Figure 64: Top 20 features using CRF on the i2b2 data set for category B-IdNumber

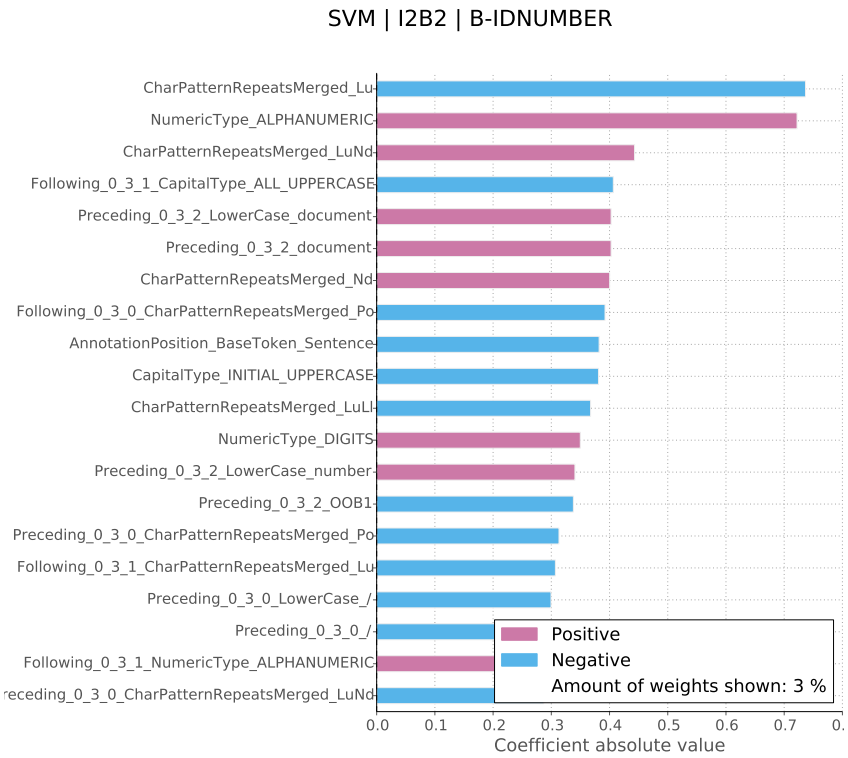


Figure 65: Top 20 features using SVM on the i2b2 data set for category B-IdNumber

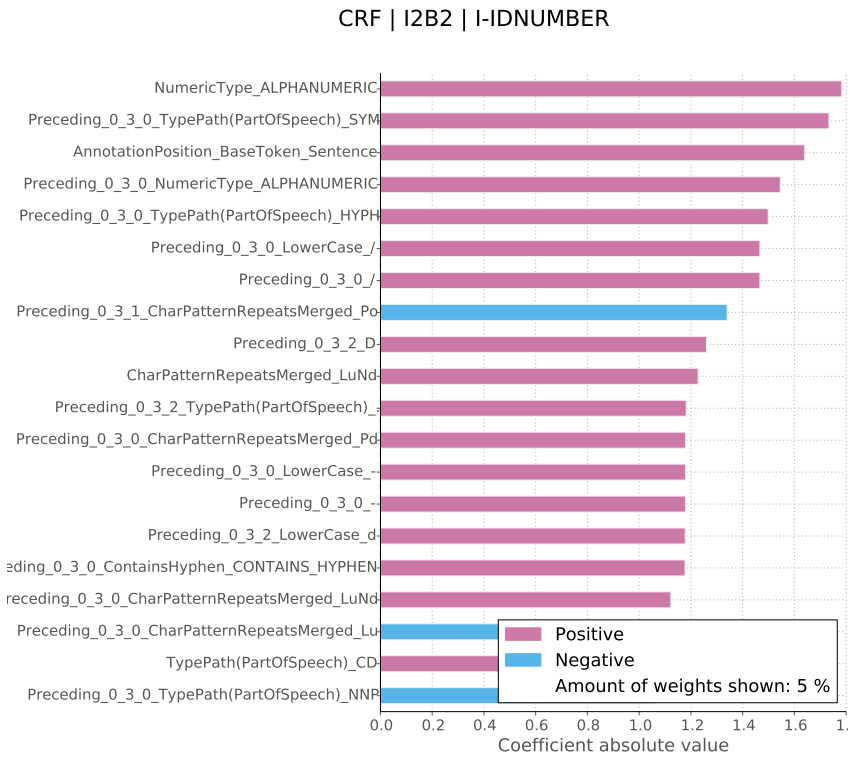


Figure 66: Top 20 features using CRF on the i2b2 data set for category I-IdNumber

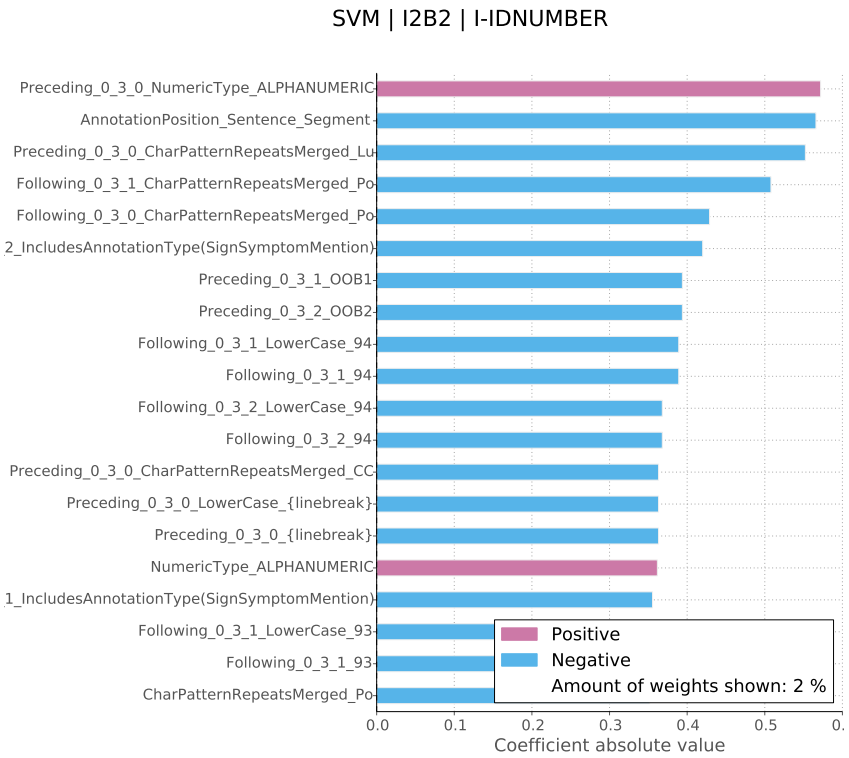


Figure 67: Top 20 features using SVM on the i2b2 data set for category I-IdNumber

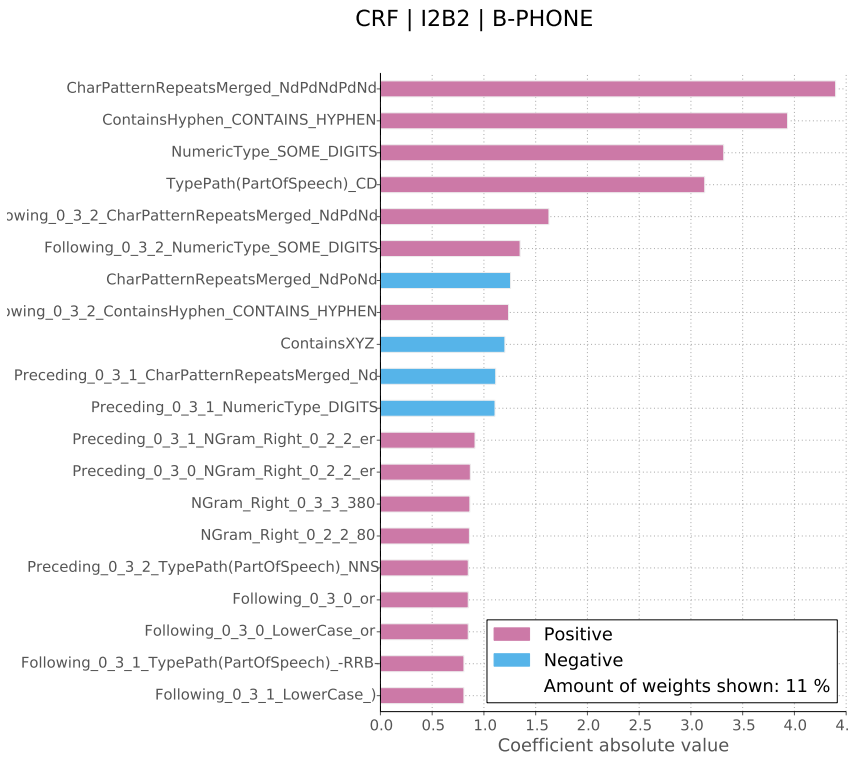


Figure 68: Top 20 features using CRF on the i2b2 data set for category B-Phone

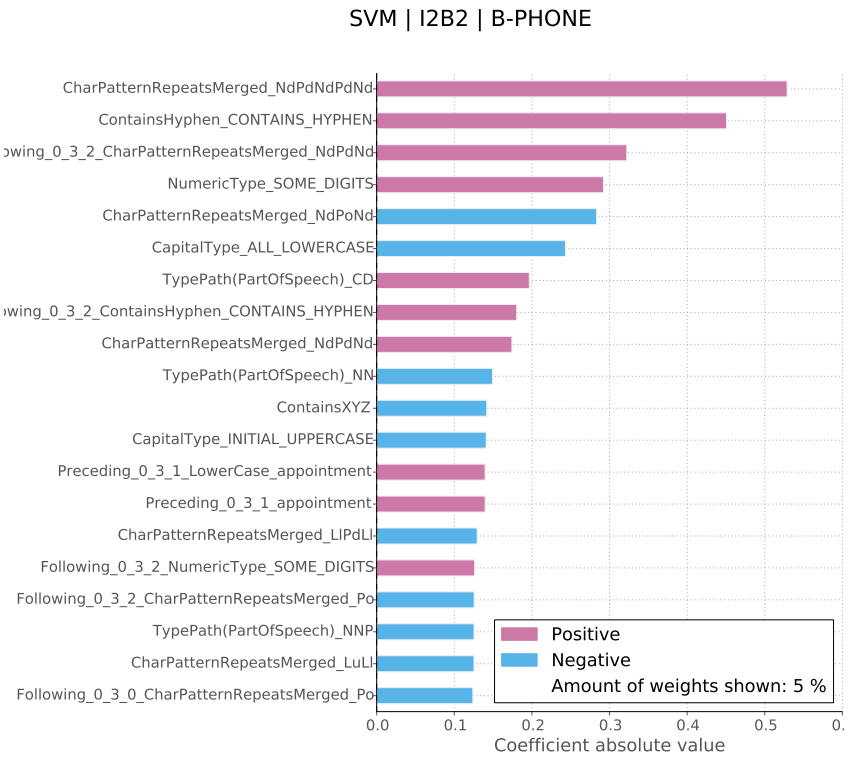


Figure 69: Top 20 features using SVM on the i2b2 data set for category B-Phone

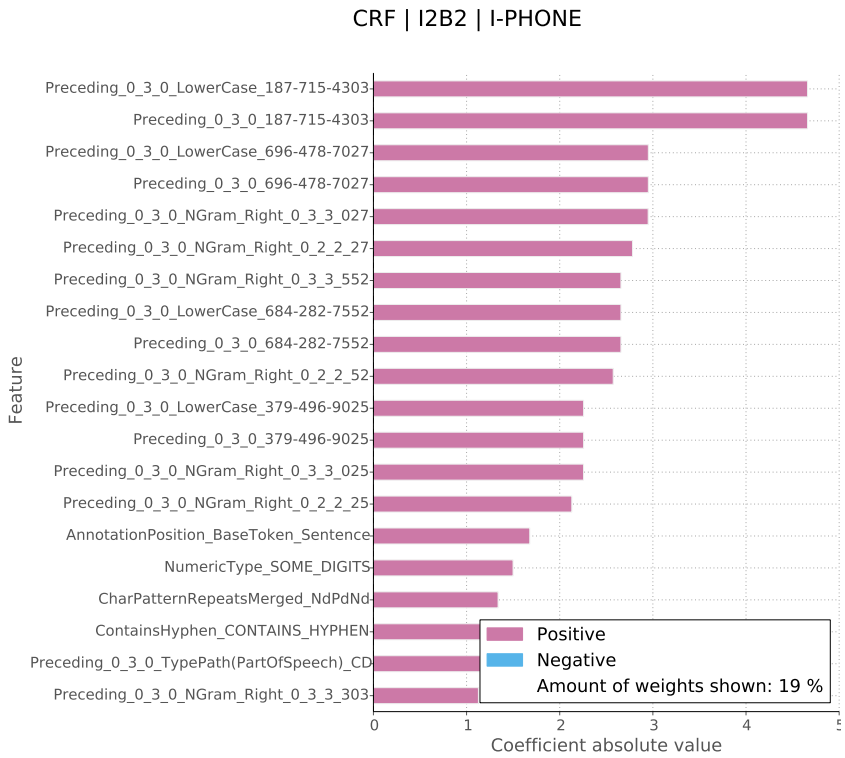


Figure 70: Top 20 features using CRF on the i2b2 data set for category I-Phone

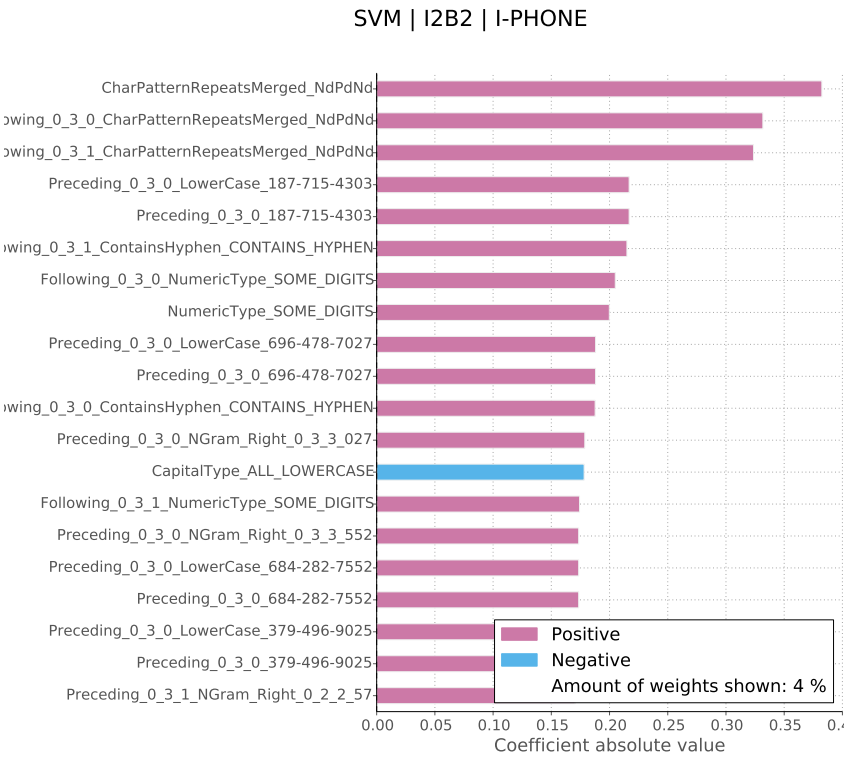


Figure 71: Top 20 features using SVM on the i2b2 data set for category I-Phone

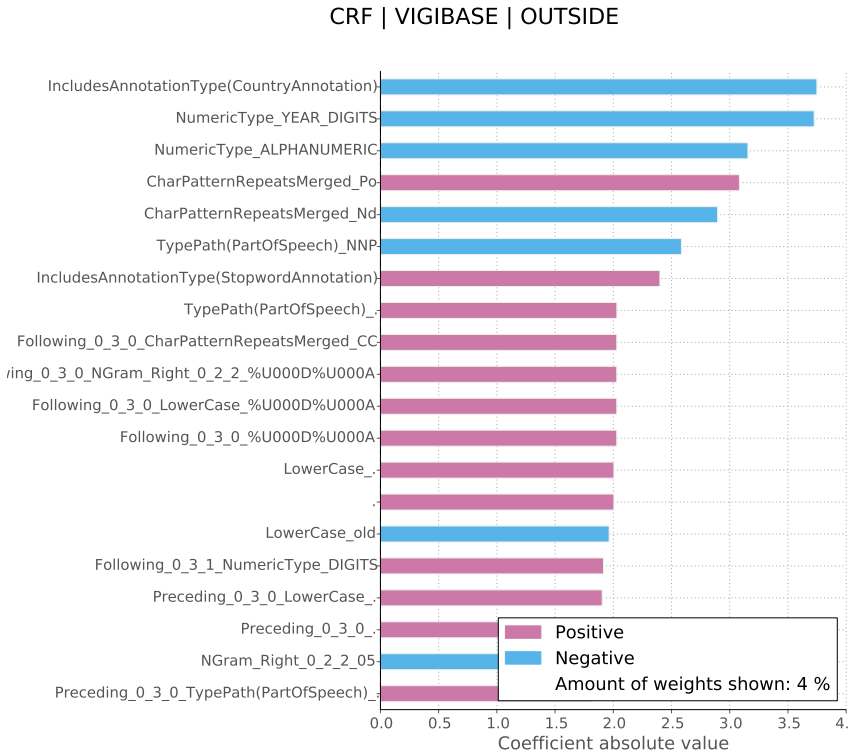


Figure 72: Top 20 features using CRF on the VigiBase data set for category Outside

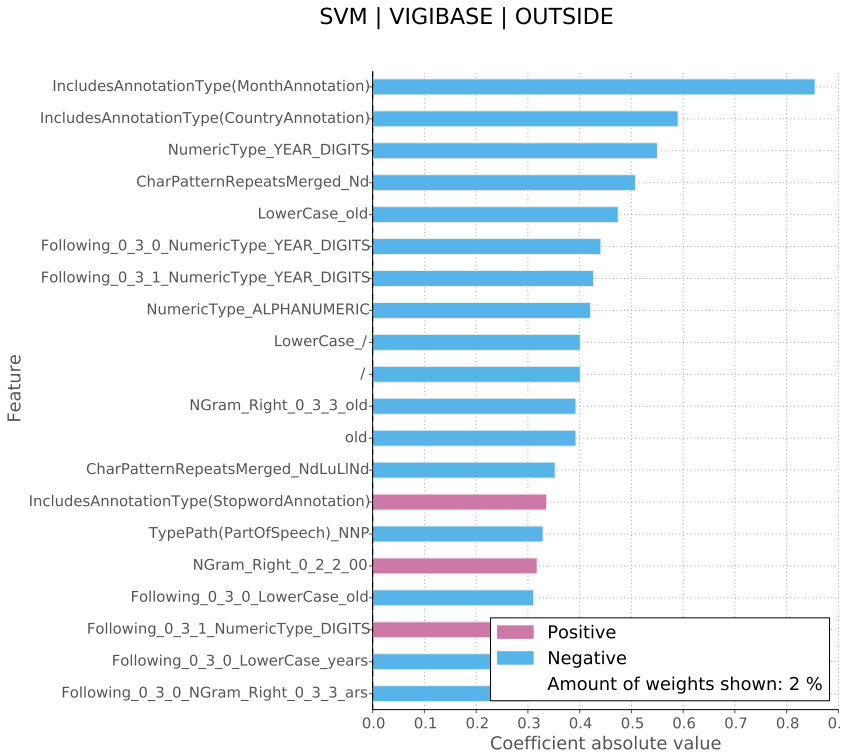


Figure 73: Top 20 features using SVM on the VigiBase data set for category Outside

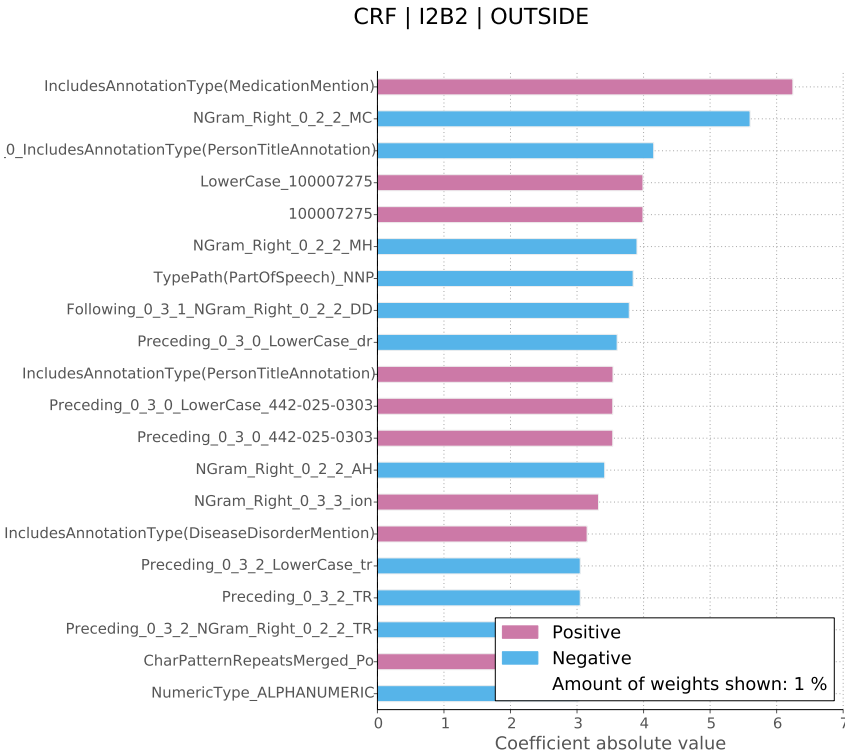


Figure 74: Top 20 features using CRF on the i2b2 data set for category Outside

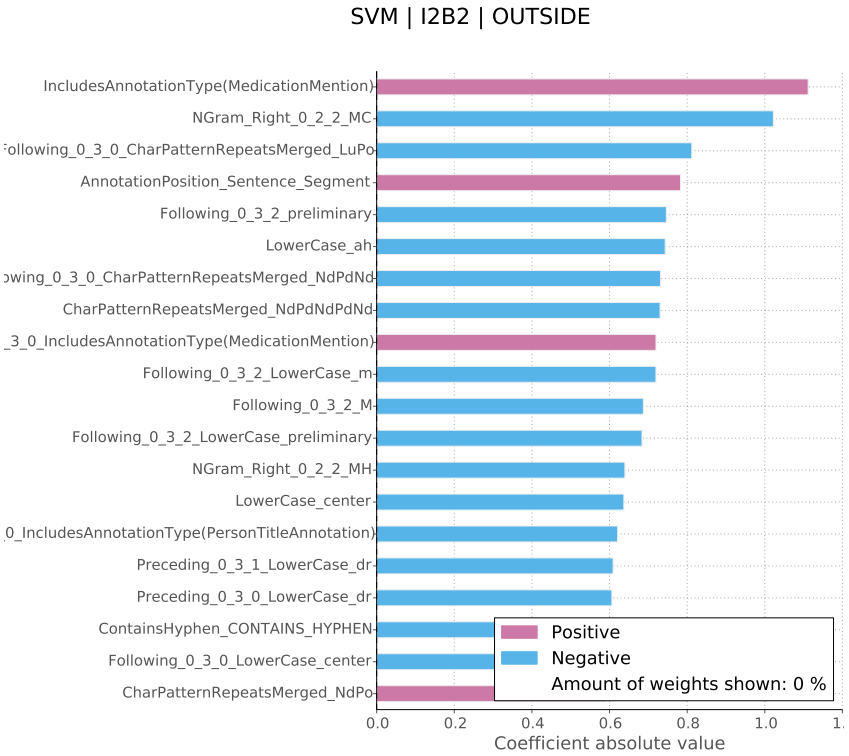


Figure 75: Top 20 features using SVM on the i2b2 data set for category Outside