Text Mining In Insurance: From Unstructured Data To Meaning

Diego Zappa[1], Mattia Borrelli[2], Gian Paolo Clemente[3], Nino Savelli[3]

[1] Department of Statistical Sciences, Università Cattolica del Sacro Cuore – Milan, e-mail: diego.zappa@unicatt.it, *Corresponding Author*
[2] Actuarial Analyst, Swiss Re, e-mail: mattia_borrelli@swissre.com
[3] Department of Mathematics, Finance and Econometrics, Università Cattolica del Sacro Cuore – Milan e-mail: gianpaolo.clemente@unicatt.it, nino.savelli@unicatt.it

ABSTRACT: Every day insurance companies collect an enormous quantity of text data from multiple sources. By exploiting Natural Language Processing, we present a strategy to make beneficial use of the large information available in documents. After a brief review of the basics of text mining, we describe a case study where, by analyzing the accident narratives written by the researchers of the National Highway Traffic Safety Administration (NHTSA) of the U. S. Department of Transportation, we aim at grasping latent information useful to fine-tune policy premiums. The process is based on two steps. First, we classify the reports according to the relevance of their content to find the risk profile of the people involved. Next we use these profiles to add new latent risk covariates for the ratemaking process of the customers of a company.

KEYWORDS: Text mining, N-Grams, Natural language processing, policy premiums

1. Introduction

Text mining is a knowledge-intensive process in which the analysis is represented by documents (Bash, 2015). Originally introduced just for descriptive purposes, recently it has quickly evolved by adding methods able to classify documents according to their latent topic or to infer about the "sentiment" of customers or the users of social networks. The boost to these approaches has gone along with the evolution of both the computational efficiency of the algorithms necessary to analyze textual data and the technology needed to store information.
This work does not aim at explaining in detail what text mining is. Relevant and comprehensive references are provided by Bash (2015), Feldman and Sanger (2006). What we want to show is how

to exploit this methodology to extract valuable information for insurance companies from unstructured data. In insurance, akin to many other contexts, there is a strong need of text classifiers. This is due to the evidence that companies collect every day an enormous quantity of text data from multiple sources such as customer feedback, claims adjuster notes, underwriter notes, police reports on accidents, medical records, surveys, emails, web documents, social media posts, etc. Text analysis could help to refine:

- marketing campaigns
- brand management
- fraud detection and other complaints
- claims management and compensation
- subrogation
- relationships between the help center and clients
- analyze contracts' clauses

To have an idea of the possibilities offered by text mining, a UK insurer[1] recently introduced a motor insurance policy aimed at granting discounts to first time drivers who agreed to give the company access to their Facebook profiles[2] (recent data privacy rights do not allow to repeat the same experiment). Given access to the profile, the company developed an algorithm that analyzed all user posts, likes and GPS locations in order to perform a personality test. By analyzing the style of writing of each user, the algorithm could uncover positive and negative traits. In this way, personality traits can be used as predictors of a customer's life and driver behaviour and to assess a criterion for determining eligibility for an insurance discount.

Therefore, one of the potential competitive advantages of text exploration in insurance is the possibility of enriching the customer risk-profiles based upon the standard structured "Company DB" customer database. Ideally, we are referring to a framework in which actuaries gather information from an unstructured "Cloud DB", fed by an external source (e.g., documents, web sources, etc.). The Cloud DB shares information with the Company DB through appropriate link variables able to link the Cloud DB profiles and the customers of the company, thereby allowing the aggregation of the two apparently disjointed data sources.

Text mining is a challenging research field. Issues range from the need to analyze very large quantities of data, the unstructured nature of text data and the complexity in finding keys to standardize language for inferential purposes. For example, in the case of insurance companies, text data varies from colloquial to formal language. In police reports or claims adjuster notes we might find that the terminology used is often repetitive. Lexical structures are sometimes fixed and predictable, but this does not apply to social media, which is always changing and therefore cannot be analyzed using standard methods.

This contribution fits into the big data paradigm (Bühlmann *et al.*, 2016). Generally speaking, big data may be depicted as an unstructured, large, heterogeneous and unstable dataset that often hides latent relevant information not measurable through a standard sampling process. Among others characteristics, big data may refer to documents, the flow of tweets on the web, any social network, sentiment about the health of the economy, the status of either a country or a company, or the flow of documents produced during daily work (e.g., reports, recipes, phone calls, mails). By

[1] See: https://www.telegraph.co.uk/insurance/car/insurer-trawls-your-facebook-profile-to-see-how-well-you-drive/

[2] See: https://www.telegraph.co.uk/technology/2016/11/02/facebook-blocks-admiral-app-that-priced-car-insurance-based-on-s/

an extensive description of a case study, we aim at showing the possibilities given by text mining to grasp latent useful information for insurance that might be used to fine-tune policy pricing or to better assess the customer risk profiles.
The paper is organized as follows. Section 2 briefly describes how a document can be analyzed by using text mining. Section 3 shows how natural language processing (NLP) (Clark *et al.*, Fox and Lappin, 2010) algorithms may be used to classify a document. Section 4 tests the efficacy of text mining in grasping latent information useful for insurance pricing and presents results of NLP applied to a collection of reports about accidents that occurred in the United States between 2005 and 2007. The narratives were produced by the National Highway Traffic Safety Administration (NHTSA, 2008) researchers sent to crash scenes. Researchers listened to police scanners, interviewed everyone on the scene and then filed a separate standardized report. The reports collected structured data, like date and time of accident, whether/road conditions, driver use of medication, driver use of a cell phone. Then they wrote a brief report, describing the accident in a maximum of 1,200 words[3]. Conclusions follow.

2. N-Grams and the prediction of words

Documents are the focus of text mining. A collection of documents refers to a corpus. Once we have chosen the document unit, we need to set the granularity level of the analysis, after which we can analyze single characters, words, phrases, even groups of phrases.
One of the most common tools used to analyze a document is N-gram, a sequence of n words (or even of characters) obtained from a document. This is a sort of rolling window of size n: by moving this window by one position at the time, we obtain a list of new N-grams. For example, let us consider the sentence "*The police stopped a vehicle without insurance*" . We can build different N-grams by varying the length n:

n = 1 (**unigram**) returns {"*The*", "*police*", "*stopped* ", "*a*", "*vehicle*", "*without* ", "*insurance*"}
n = 2 (**bigram**) returns {"*The police*", "*police stopped* ", "*stopped a*", "*a vehicle*", "*vehicle without* ", "*without insurance*"}
n = 3 (**trigram**) returns {"*The police stopped*", "*police stopped a*", "*stopped a vehicle*", "*a vehicle without* ", "*vehicle without insurance*"}

The choice of the order of the N-grams is linked to the complexity of the problem and to the scope of the analysis. Using a high value of *n* means incorporating more context in the units of the document, while a low *n* value means that the basic unit of data is going to be more granular.
An N-gram model can also be seen as a *Markov chain* of $m - 1$ elements, where m is the length of the sentence (e.g., number of words). Therefore, N-grams might be seen as a stochastic process that predicts words given a certain history or context. Since it is a Markov chain, it does not take into account all the history of the previous words but considers only the most recent word/history. This

[3] At the time of this paper submission, the NHTSA database was available at https://crashviewer.nhtsa.dot.gov/LegacyNMVCCS/Search. It out of scope of this paper to give details about it. We remind only some relevant goals of the NHTSA research: understanding the pre-crash environment, the investigation about the causes of the rollover problem, the description of the traffic environment in which the crash occurred, the behavior of persons and vehicle involved, the specific outcomes, the drugs impairment etc.

simplifies how documents are written because it does not focus on grammar rules to estimate words but only on their context. According to this framework, we can build the joint probability of words in a sentence using the Chain Rule:

$$P(w_1 w_2 \dots w_m) = \prod_{i=1}^{m} P(w_i | w_1 w_2 \dots w_{i-1}) \tag{2.1}$$

where $w_i, i = 1, , \dots, m$ is the i-th word of a sentence of length m and equals $P(w_1)$ when $i = 1$. It is worth noting that we obtain the joint probability of a document by multiplying the probability of each word conditioned over all the previous words. For the sentence reported above, we obtain:

P ("*The police stopped a vehicle without insurance*") =
P (*The*) × *P* (*police*|the) × *P* (*stopped*|The police) × *P* (*a*|The police stopped)×
P (*vehicle*|The police stopped a) × *P* (*without*|The police stopped a vehicle)×
P (*insurance*|The police stopped a vehicle without)

To compute the probability, this approach requires counting and dividing the occurrences for all the possible sentences. An example is:

$$P\ (vehicle|\text{The police stopped a}) = \frac{\#(\text{The police stopped a vehicle})}{\#(\text{The police stopped a})}$$

Unfortunately, this will lead nowhere because we will hardly have enough data to compute these probabilities consistently. To avoid this issue an N-Gram *language model* is used. The aim is to limit the context in which a word is used. Instead of using all the previous word history, we only take a subset of it. Therefore, the (2.1) can be approximated using an N-gram model of order *n* according to:

$$\begin{aligned} P(w_1 w_2 \dots w_m) = \prod_{i=1}^{m} P(w_i | w_1 w_2 \dots w_{i-1}) \cong \prod_{i=1}^{m} P\left(w_i | w_{i-(n-1)} \dots w_{i-1}\right) \\ = \frac{\#(w_{i-(n-1)} \dots w_{i-1}, w_i)}{\#(w_{i-(n-1)} \dots w_{i-1})} \end{aligned} \tag{2.2}$$

where the number of words used to condition the probabilities is *n* − 1.
For instance, in the case of a **bigram** language model, each word probability will be conditioned only by the previous word. In our example we have

P ("The police stopped a vehicle without insurance") =
P (*The*) × *P* (*police*|The) × *P* (*stopped*|police) × *P* (*a*|stopped)×

$$P(vehicle|\text{a}) \times P(without|\text{vehicle}) \times P(insurance|\text{without})$$

The use of an N-gram model allows us to reduce the number of cases in which occurrences are counted. Intuitively, as the order of the N-gram model increases, the N-gram frequency decreases. This is because every character sequence obtained from the model needs to be matched over the entire corpus (for example, "*stopped a vehicle"*).

The previous description is an oversimplified approach to document analysis. In every text mining application, a key point is to understand how words and punctuation are joined together to express concepts. The definition of grammar rules is certainly crucial, but to assure a proper representation of data we need to simplify these issues.

The "*bag-of-words*" is an approach used to deal with the complexity of text data. It assumes that all the terms have the same importance in a document; that is, there is no distinction between different parts of speech (verbs, nouns, adjectives, etc.). In this way, we are not interested in their position in the text, and so they can be seen as a set of strings without meaning. The term "*bag-of-words"* is self-explanatory because it refers to a document as a bag of words that can be extracted without considering the order.

2.1 Representing text data

To show how a document can be numerically represented, let us consider, for instance, the following two simple phrases (which will be generically called "documents"):

1. *Autonomous car insurance. Look, no claims!*
2. *Self-driving cars are set to change motor insurance radically*

Next, let us consider a matrix on whose columns we place every unique term in the corpus and on whose rows the document ID*s*. When terms are present/not present in documents, the cells will take the value 1/0, thereby giving the matrix a *non-negativity* connotation.

Table 1: Numerical representation of documents

		Terms											
		Are	Autonomous	Car	Cars	Claims	Change	Driving	Insurance	Look	Motor	No	Self
Document	**1**	0	1	1	0	1	0	0	1	1	0	1	0
	2	1	0	0	1	0	1	1	1	0	1	0	1

This representation allows us to move from unstructured data (i.e., a collection of text documents) to structured data that can be analyzed by applying data mining methods. This numerical structure also has three important properties: *Sparsity, Non-negativity* and *Side information.* Punctuation also plays an important role in giving context to documents, especially when we have corpora composed of documents retrieved from the Internet, emails and other sources of data where an informal language is typically used. We can add the frequency with which signs are present among classified documents. This will give us an indication on how frequent some punctuation signs are in documents of every type (happy, positive, sad, negative and so on . . .).
To obtain a simpler numerical representation of the corpus, we can opt to reduce the number of features, e.g., prepositions, "generic" verbs ("be", "have"), etc. In Table 1 we have eliminated duplicated terms, punctuation, derived words and so on. These are a few of the dimensionality reduction techniques applied in the following paragraphs to reduce the complexity of document representation.

2.2 Tokenization

Tokenization is a process that consists in breaking up documents (classified as word, N-grams or phrases) into elements called tokens, which are used as an input to text mining procedures.
Tokens might be similar but stylistically very different from one another: for example, some have capital letters while others have numbers or punctuation signs in them. Other than that, some tokens have no meaning and thus would add no information to the documents' representation. To solve these issues, we apply the most used normalization techniques to the tokens to obtain the most meaningful subset of them. Examples of non-relevant tokens are: case-folding, punctuation and numbers, hyperlinks and spelling.
A relevant case is given by American and British English spelling. Despite the fact that both fall under the English language category, there are cases where the spelling of certain words is done differently:

- "*Analyze*" - "*Analyse*"
- "*Program*" - "*Programme*"
- "*Center*" - "*Centre*"

In an information extraction problem this causes difficulties: even if the words have the same meaning, the fact they are spelled differently prevents us from obtaining matches during search operations. Obviously, the same holds for words that are written differently in the two *dialects*: "*holiday*" or "*vacation*", "*hire purchase*" or "*installment plan*", and so on.
Once the tokenization process is completed, we reduce the number of features to select the more significant ones by removing the so-called *stop words*. We can think of *stop words* as words that are useful for the syntactic construction of phrases but whose semantic value is important relative to the context in which they are inserted. For example, the use of the prepositions "to" or "for" can change the interpretation of phrases ("*I'll take it to him*" and "*I'll take it for him*" have a different meaning). Thus, in this case removing them might affect the interpretation of the phrases and undermine the document's meaning. To deal with stop words we use a *stop words dictionary*, which is a collection of all the terms we consider unnecessary to our specific problem and thus can be

removed. Since stop words dictionaries are already available in all of the most common languages, a dictionary can be downloaded and then tweaked by adding or removing specific terms.

2.3 Stemming and lemmatization

Once the removal of numbers and punctuation has been completed, we can further simplify the features using a stemming or a lemmatization process. In English, as well as in other languages, verbs have different inflectional forms or suffixes to which they are related. For example:

a. *see, saw, seen → see*

b. *insurance, insurer, insure, insured → insur*

Stemming is merely a heuristic process that truncates the end of every word to reduce it to its common base, also called *root*. In English a process that truncates the last characters of every term may have success in removing excess characters, leaving only the common base for the matching algorithms. The most common *stemming algorithm* for the English language is the one proposed by Porter (1980).
To improve the accuracy of the truncation process we can use another technique called *lemmatization*. Basically, instead of defining rules to truncate words, we use a lemmatizer, which carries out a full morphological analysis to accurately identify the lemma for each word. Lemmatization removes inflection suffixes and compresses words into the so-called "lemma". We define a lemma as the canonical form of a term, deprived of most conjugation suffixes and transformations. Intuitively, we can think of a lemma as the word we look up in a dictionary when searching for a specific term.

An example of text reduction with the Twitter corpus

To show the impact of these processes, we refer to a corpus of documents collected from Twitter. We have retrieved tweets regarding the insurance sector in Italy. The tweet gathering process was spread across three months, making it possible to collect 15,909 tweets sent by persons that mentioned the word "*Insurance*". The number of documents in this corpus is far greater than the number of documents used to explain how to prepare a text before the representation. Table 2 emphasizes how the processes mentioned above are able to reduce the document complexity.

Table 2: The impact of normalization and tokenization on a corpus of tweets

Process	Tokens
Original data	237,091
Tokenization	34,770
Token normalization	15,051
Stop words	14,847
Stemming	10,878

The original corpus of 15,909 documents is composed of 237,091 tokens. This number indicates the total number of words from the tokenization process. Note that after the stop words the size of the tokens was reduced to 6% of the original data.

2.2 Vector Space Models

Once the list of tokens is ready, we need to find a way to describe the whole document. One choice is to use Vector Space Models (VSM) (Lappin and Fox, 2015, chp.16).
A document can be seen as a vector whose dimensions are given by the number of features (Turney and Pantel, 2010). This representation is called "Document Term Matrix". By switching columns and rows, we obtain the so-called Term Document Matrix**.**
Placing documents into a multi-dimensional space requires an accurate coordinate system. Ideally, documents that are similar are also close to each other, while documents that are semantically different need to be distant. Using only term frequency is not enough to capture similarities if there are no informative words with a high occurrence rate. Hence, the **tf-idf** (*term frequency–inverse document frequency*) technique is typically used; this is a statistic that reflects the importance of terms in a corpus.
In general, a term is more important than others when it occurs multiple times in a document and when it is also rarer than other terms in the corpus. If we are analyzing a collection of documents coming from a division of an insurance company, we may find that terms such as *claims, underwriter, premium* and *reserve* appear quite frequently. However, their presence is common to every document in the collection, and so the value of the tf-idf statistic will not be as high as we would have expected while using only term frequency. Terms such as *windstorm, subrogation* and *tsunami* may be less frequent than the previous ones but they are more peculiar and rarer. Peculiarity and rarity are two qualities we look for when representing documents. To this end, the *td-idf* statistic combines two quantities:

1. term frequency

2. document frequency

Term frequency

Term frequency is the frequency of the term t in the document d. It is the result of a simple tabulation process of the document text

$$tf(t,d) = f_{t,d} = \frac{{}_dT_t}{T} \quad (2.3)$$

where ${}_dT_t$ is the number of occurrences of t in d, while T is the total number of terms in the document. One problem with term frequency regards long documents. Longer texts have a high probability that some words are going to be repeated, thereby leading us to conclude that the term frequency of these words is higher than what we would expect in shorter documents. To mitigate this effect, we can use a normalized term frequency that includes the use of a smoothing parameter based on a combination of a weighting parameter and a damping function: $ntf(t,d) = a + (1-a)\frac{tf(t,d)}{tf_{max}\,(t,d)}$, where $tf_{max}\,(t,d)$ is the maximum term frequency of all the terms in the document d and a is the smoothing parameter that can assume values between 0 and 1 (typically around 0.4-0.5).

Document frequency

Document frequency indicates the *inverse document frequency* of the term *t* in the collection of N documents. It is equal to the log of the ratio between the number of documents in the collection, N, and the number of documents N_t in which the term *t* appears:

$$idf(t,N) = \log\left(\frac{N}{N_t}\right) \quad (2.4)$$

This quantity describes how many rare terms there are in the corpus; thus, in the presence of a rare term, (2.4) assumes a high value; otherwise, when many documents share the word, (2.4) takes on a small value.

Term frequency - Inverse document frequency

Ultimately, we can compute the *tf-idf* statistic for each document by simply multiplying the *term-frequency* and *inverse document frequency* as follows:

$$tf - idf(t,N) = tf(t,d) \times idf(t,N) \quad (2.5)$$

tf-idf assumes a high value in the case of a high term frequency and a low frequency in the collection. To better explain how the term frequency-inverse document frequency works, we use the following five documents:

a. *Autonomous car insurance. Look, no claims!*

b. *Self-driving cars are set to change motor insurance radically*

c. *Insurers agree that, reputationally, their brand image is often made or broken during the claims process*

d. *Industry executives broadly agree that advanced analytics can be used to drive value in insurance*

e. *Progress has been slower in other lines of business, such as general liability, most specialty lines, and other elements of life insurance*

We proceed by building a Term Document Matrix using the tf-idf statistic. Table 3 shows how common stems such as ”insur”, “claim” and ”agre” have low tf-idf values because of their presence in multiple documents. The statistics enables us to describe documents by considering peculiar terms and avoiding a representation based on common terminology. In fact, the stem ”insur” has value 0 - despite its high frequency, the highest among all the stems - meaning that it will not affect at all the representation of the corpus. Words that appear only once in a document will have a high tf-idf value and influence the position of the document in the multi-dimensional space. [4]

[4] To compress further the table, for each document we may compute the average tf-idf statistics associated with every stem. A document with a high value means there are words in it that are on average more peculiar respect to others.

Table 3: TDM (tf-idf)

Terms	Docs				
advanc	0.000	0.000	0.000	0.211	0.000
agre	0.000	0.000	0.132	0.120	0.000
analyt	0.000	0.000	0.000	0.211	0.000
autonom	0.464	0.000	0.000	0.000	0.000
brand	0.000	0.000	0.232	0.000	0.000
broad	0.000	0.000	0.000	0.211	0.000
broken	0.000	0.000	0.232	0.000	0.000
busi	0.000	0.000	0.000	0.000	0.211
can	0.000	0.000	0.000	0.211	0.000
car	0.264	0.189	0.000	0.000	0.000
chang	0.000	0.332	0.000	0.000	0.000
claim	0.264	0.000	0.132	0.000	0.000
drive	0.000	0.000	0.000	0.211	0.000
element	0.000	0.000	0.000	0.000	0.211
execut	0.000	0.000	0.000	0.211	0.000
general	0.000	0.000	0.000	0.000	0.211
imag	0.000	0.000	0.232	0.000	0.000
industri	0.000	0.000	0.000	0.211	0.000
insur	0.000	0.000	0.000	0.000	0.000
liabil	0.000	0.000	0.000	0.000	0.211
life	0.000	0.000	0.000	0.000	0.211
line	0.000	0.000	0.000	0.000	0.422
look	0.464	0.000	0.000	0.000	0.000
made	0.000	0.000	0.232	0.000	0.000
motor	0.000	0.332	0.000	0.000	0.000
often	0.000	0.000	0.232	0.000	0.000
process	0.000	0.000	0.232	0.000	0.000
progress	0.000	0.000	0.000	0.000	0.211
radic	0.000	0.332	0.000	0.000	0.000
reput	0.000	0.000	0.232	0.000	0.000
selfdriv	0.000	0.332	0.000	0.000	0.000
set	0.000	0.332	0.000	0.000	0.000
slower	0.000	0.000	0.000	0.000	0.211
specialti	0.000	0.000	0.000	0.000	0.211
use	0.000	0.000	0.000	0.211	0.000
valu	0.000	0.000	0.000	0.211	0.000

2.2 Placing documents in a multi-dimensional space

Having represented the documents in a space, it is also interesting to compare *pairs of documents* to understand document similarities.
By using VSM (Vector space models), the closer the document vectors are, the more similar are the documents (see, e.g., Pennington et al., 2014). To evaluate the position of the documents, we use a measure called cosine similarity, a statistic that also implies a length normalization process. Given only two documents, let $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ be the two column vectors containing the tf-idf values for the M terms in the collection. The cosine similarity is defined as

$$\cos(\theta) = \frac{\boldsymbol{V}_1 \cdot \boldsymbol{V}_2}{\|\boldsymbol{V}_1\|\|\boldsymbol{V}_2\|}$$

We report below the matrix computed by using the corpus of five documents described above. Values show that this collection includes very dissimilar documents. The small size of the collection and the limited number of stems justify the results. In this case, the use of *tf-idf* has over-weighted the search for rare terms with high discrimination abilities.

$$\begin{bmatrix} 1.000 & 0.086 & 0.071 & 0.000 & 0.000 \\ 0.086 & 1.000 & 0.000 & 0.000 & 0.000 \\ 0.071 & 0.000 & 1.000 & 0.038 & 0.000 \\ 0.000 & 0.000 & 0.038 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$$

3. Natural Language processing (NLP)

After the very demanding process of document normalization, we are ready to find statistical methods that can "naturally" process languages to extract information. We are looking for efficient statistical methods that "simulate" the process of our brain when we are reading or listening to people speaking.
The N-gram approach may fail in this task. It does not take into account the overall context of the phrase, unless the size of the N-gram windows is really wide; however, when the size of the N-gram increases, the accuracy of estimated word probabilities decreases. Supervised machine learning methods are not able to do that without a time consuming and sometimes painful calibration of the training phase.
NLP is a mix of artificial intelligence (AI), computer science and computational linguistics (CL) to extract meaning from documents, recognize text and, ultimately, model and shape text in order to compose original contents (Bowman *et al.*, 2015).
In insurance, NLP may be useful to make inferences about frauds (Mailvaganam 2005, Stout 1998), customer sentiment (Liu 2015, Ceron et al. 2016), or (see our case study) to grasp latent information that cannot be measured by standard methodologies.

A revolutionary method to process language is the so-called *word2vec*. Word2vec is not just an algorithm but a class of algorithms introduced by Tomas Mikolov *et al.* (2013a, 2013b) that contains two different models: Continuous Bag of Words (CBOW) and Skip-Gram. These algorithms are an example of deep learning (Wiley, 2016). Deep Learning is a branch of Machine Learning based on multi-layered neural networks that involve both linear and non-linear transformations. The algorithms in Mikolov et al. can be defined as two instances of a shallow neural network (with only one hidden layer). The importance of word2vec methods does not reside only in their predictive abilities. We show that they also enable us to provide a meaningful representation of words. Instead of trying to represent words in relation to their use in a document, the representation of words obtained with word2vec models concentrates on the meaning of the words themselves. For example, using the concept of "window" in the N-gram models, we define the context as the words that surround the central word "window" itself. For example, in the string "Deep learning methods", the words "Deep" and "methods" compose the context of the central word "learning ".

3.1 Continuous Bag of Words (CBOW)

The CBOW model was introduced by Mikolov et al. in 2013. We start by describing the simplest version based on a one-word context. In this version, the model predicts one word, given only one context word, to form a *bigram*. For example, given the word *insurance*, the model will try to predict the immediately following word.
To achieve this aim, we train the neural network based on the structure reported in Figure 1 (see also Rong, 2014) .

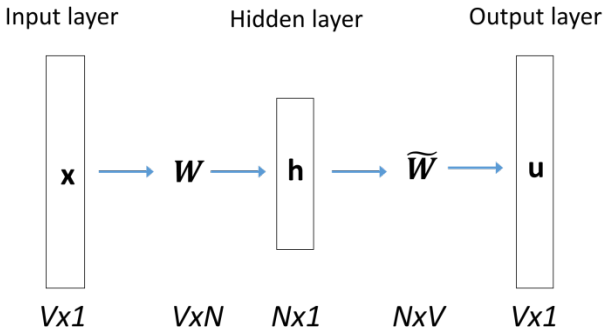

Figure 1: A simple CBOW model with only one word in the context

For each word we are interested in we use a vocabulary of size V from the corpus of documents we are analyzing. Each word will be represented by one-hot vector $\boldsymbol{x}$ of dimension V. Next, we shrink the V dimensions into a smaller space of size $N < V$ (*hidden layer*). The transition is made by using the $V \times N$ weight matrix. Each node h of the neural network is defined by the simple linear *activation* transformation $\boldsymbol{h} = \boldsymbol{x}^T\boldsymbol{W}$. The transition from the hidden layer to the output layer is performed by using the $N \times V$ matrix $\widetilde{\boldsymbol{W}}$.
These matrices provide the score $\boldsymbol{u} = \widetilde{\boldsymbol{W}}(\boldsymbol{x}^T\boldsymbol{W})^T$ computed for every word in the vocabulary.

Let $\boldsymbol{x_I}$ be the input vector (for instance, the word "*insurance*") and suppose we want to maximize the probability $P(\boldsymbol{x_O}|\boldsymbol{x_I})$, where $\boldsymbol{x_O}$ is the word we expect to find next to $\boldsymbol{x_I}$ (say "*policy*"). Once we

have computed an initial score $\boldsymbol{u}$ by using the stochastic gradient descent (Guelman, 2012), it is possible to update the weights of $\boldsymbol{W}$ and $\widetilde{\boldsymbol{W}}$ until convergence. The scores $\boldsymbol{u}$ say what words within *V* are closer to $\boldsymbol{x}$**.**

One of the advantages of the word2vec algorithms is that $\boldsymbol{W}$ is also used to obtain a denser and more meaningful representation of the words in input. From a practical point of view, the N dimensions of $\boldsymbol{W}$ represent the word similarity, i.e., the N words that most likely are expected to be close to the words used in the training phase. Using the notion of *cosine similarities* described in section 2.4, we can compute the similarities between words and obtain additional information about terms and their semantics.

Another property of word2vec models regards "clustering themes", or the ability of these models not only to find similar words but also to discover a class of items regarding similar topics. These models do not typically perform topic modeling, but practical evidence shows that a topic can be assigned to similar word vectors. Therefore, terms that appear in the same context can be interpreted as observations of a latent topic distribution over words.

Hence, word2vec models can be useful in discovering semantic relationships between words. This is one of the most important features of this class of algorithms, allowing us to overcome the simple similarity between words. Similarities between groups of words, known as linguistic regularities (Mikolov et al., 2013), can indeed be detected.

4 Insurance case study

We analyzed 6,949 police reports (written by NMVCCS researchers) on accidents in the United States between 2005 and 2007.

In the literature, an example of text mining application to this database was proposed by Borba (2013, 2015). From documents he extracted dichotomous variables (called flags) to study the impact on accidents of weather conditions, the status of the cars, the usage of mobile phones while driving, the dynamics and the locations of accidents, the driver's condition (presence of driver fatigue, use of cell phones, medications, drug, alcohol, prescriptions).

Our purpose is to classify documents not only according to "accident specific" keywords. In particular, we look for information that can uncover new risk covariates which might be used to fine-tune policy pricing or to improve customers' risk-profiles. To achieve this aim, a straightforward application of N-grams may be misleading. For example, a key issue is the profile of drivers on the basis of the substances reported in the policy reports, if available. The difficulty here is that, on the one hand, the words "medication", "drug", "prescriptions", "alcohol" are not always explicit in police reports, and on the other some substances could belong to more than one category. In addition, some verbal expressions such as "*he was not taking medication*" or "*the driver was aggressive but the BAC results were negative*" could easily lead to the wrong classification. Therefore, without a supervised check by an expert, the N-gram approach could fail. To avoid these pitfalls, we have improved the analysis of documents by training an NLP, based on a word2vec algorithm, to "*automatically"* classify the substances. In what follows we will focus mainly on this case, but the same procedure can be applied to other contexts, such as driver-related factors (distracted, using a cell phone, …).

The procedure we used to analyze the dataset is based on the following 6 steps:

0) Obtain the dataset

1) Apply a text normalization procedure

2) Select substances (word2vec)

3) Tag part-of-speech

4) Filter off-topic cases

5) N-Gram structure and flag creation

6) Apply the prediction model

Obtaining the dataset

Unfortunately, at least at the time of the present paper's submission, the dataset containing the accident narratives cannot be directly downloaded in a user-friendly format. The dataset is published online where it can be queried using the search form created by the NTHSA. Each case is stored in an XML file that eventually can be downloaded. The peculiar trait of the XML language is that it can encode any piece of document. Indeed, being a markup language, *tags* can be created to host new and different types of information. An example is reported in Figure 2, which shows how information regarding driver fatigue during an accident can be encoded.

```
1  <FATIGUE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
2    <TRIP_LENGTH formCat="2">0.12</TRIP_LENGTH>
3    <TIME_ELAPS_DRIVING formCat="2" sasCode="1" value="1" attrCat="-9999" attrCatStr=
         "">&lt; half hour</TIME_ELAPS_DRIVING>
4    <TRIP_START_TIME formCat="2">0555</TRIP_START_TIME>
5    <SLEEPSTART_TIME formCat="2">2000</SLEEPSTART_TIME>
6    <SLEEPEND_TIME formCat="2">0430</SLEEPEND_TIME>
7    <LASTSLEEP formCat="2">8.5</LASTSLEEP>
8    <SLEEP24HRS formCat="2">8.5</SLEEP24HRS>
9    <HOURSAWAKE formCat="2">1.5</HOURSAWAKE>
10   <AVGSLEEPINT formCat="2">8</AVGSLEEPINT>
11   <SLEEPROTATE formCat="2" sasCode="1" value="1" attrCat="" attrCatStr="">No</
         SLEEPROTATE>
12   <WORKSHORTEST formCat="2">2</WORKSHORTEST>
13   <WORKLONGEST formCat="2">9.5</WORKLONGEST>
14   <WORKAVG formCat="2">4.5</WORKAVG>
15   <WORKTOTAL formCat="2">18.5</WORKTOTAL>
16   <DRIVER_FATIGUE formCat="2" sasCode="2" value="2" attrCat="-9999" attrCatStr="">
         Driver not fatigued</DRIVER_FATIGUE>
17 </FATIGUE>
```

Figure 2: Example of an XML file

Given the structure of the XML language, each case is stored differently. The quality of the information collected depends on how detailed the report was written. Each request made from the interface is passed to the *server* containing the database using the HTTP Method GET. The result of the GET request is a web page that reads the underlying XML code of the selected case and

displays it nicely.
Once we have downloaded the XML file of all the cases, we need to *parse* them. *Parsing* means that we have to understand the information encoded in the XML language. The objective of the procedure is to extract the information encoded in *sections* or *tags* and store them in a structured form. An example of an accident narrative we have extracted is:

This single vehicle crash occurred just before dawn during dark hours on a level, dry, left-curve asphalt six lane divided interstate with guardrail's in the median and outside the right shoulder. This interstate runs east and west at this location and has three lanes in each side with no traffic controls. Posted speed limit 105 kmph (65 mph).

Vehicle one (V1), a 1997 Ford Escort sedan driven by a 31 year-old female, was traveling westbound in lane two. V1 drove into the number one lane and off the right side of the road onto a paved shoulder. V1 then impacted a guardrail with its front and rebounded back onto the road re-entered all three lanes and ran off the left side of the road impacting another guardrail with its front. After the second impact, V1 rotated counter clockwise and came to rest on its wheels facing southeast in the median. V1 was towed from the scene.

The critical pre-crash event for V1 was coded: this vehicle traveling, off the edge of the road on the right side. The critical reason for the critical event was coded to V1 as a driver related factor: internal distraction. The driver stated that she was looking to the right and reaching for her cigarettes and lighter on the front seat. **The driver was in a crash three weeks prior to this crash and was taking the medications: ibuprofen, vicodin and ultram**. For this trip she was on an errand going from a coffee shop to her mother's house.

Figure 3: An example of an accident narrative.

Text normalization

In order to simplify the text mining approach, we divided each document into phrases by utilizing the dot as a string separator. The corpus size increases from the initial 6,949 document cases to 198,480 phrases. The next step in the normalization process consists in polishing up the phrases by removing punctuation signs, symbols and numbers. We have also removed the stop-words during the normalization process.
For example, the last but one phrase (in bold in Figure 3) will be changed into "*driver was crash three weeks prior crash was taking medications ibuprofen vicodin ultram*".

Selection of substances

Once we have normalized the text, we focus on the identification of substances (medications, prescriptions, narcotics and alcohol) which will be used to filter the on-topic cases that will be analyzed later. Instead of creating a list of possible substances, we opt to use word2vec methods. We were indeed able to train a neural network to predict the context words given a central word as input. This representation allows us to grasp word similarities and to uncover thematic clusters in the text.

In our particular application, we started by finding the terms that were more similar, in terms of cosine similarity, to common illegal substances (cocaine, heroin, and so on). We filtered out the corresponding names and collected them in a list. This list is used to calibrate the classification of other drugs, illegal and not. We proceed to refine the list by searching for the term closer to the term “medication” and “narcotic” to identify substances recorded during the accidents. In this way, 290 (legal and illegal) substances were “automatically” detected. For instance, in Figure 4 we report a visual representation of the results obtained by using the similarity scores between “medication” and “narcotic”. Given a word in the list, the higher of the two scores allows us to define the classification as belonging to either the “narcotics” or “medication” group. We can see how the substances are correctly separated based on their similarity scores. Just for the sake of interpretation, for example the word “medication” has a medication score of 1 since the two words are equal. The same word has a 0.6 score with narcotics: this means that, within the corpora of police reports, there are instances where the word “narcotic” is located near the word “medication” contributing to determine the high narcotic score for the word medication. In general the drugs names are pronounced closer (but not exclusively) the word narcotics, while prescription substances are more likely (but not exclusively) closer the word prescription.

Based on the texts used for the training, if a word is placed above/below the diagonal it means that it is more similar to the word placed on the y/x axis.

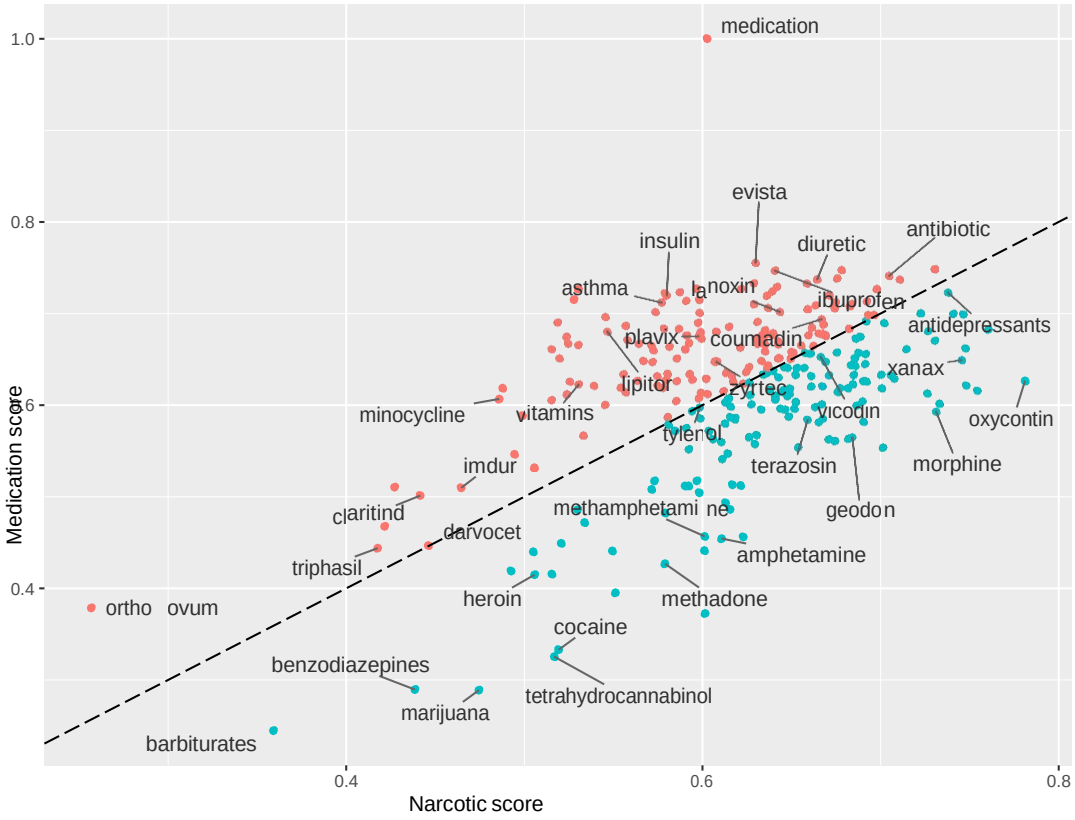

Figure 4: Drug vectors plotted by their similarities to the terms “narcotic” and ”medication”

It is worth mentioning that the scores computed by the word2vec model are *not* representative of the real similarities between any kind of drug. Thus, we should not interpret the scores as a direct representation of the real classification of each substance. Instead, word2vec creates a word embedding for each term of the specific corpus. The word embedding depends on the documents used in the training process, and thus substance discrimination reliability relies only on accurate and precise documents.

The previous result can be used in the reverse order: in general, even if a word is not mentioned in a document but occurs often in other documents when other medication/substances/narcotics substances are mentioned, then it inherits some degree of similarity to the reference group in question. For example, the word “*xanax*” never appears closer to the term “narcotics”, but it occurs frequently in phrases containing the term “heroin”. Therefore, based on the transitive property, the term “xanax” will be considered more likely to belong to narcotics than to the medications group. The extreme case is when no substance is reported: by using word similarity we can classify a document into one of the categories we have chosen.

The bottom-line for using the word2vec model is that we have created an efficient method to discriminate illegal and legal substances *with little to no effort*. In fact, the ideal word2vec model is the one that uses millions/billions of documents (for example, the entire Wikipedia library) to understand word similarities or thematic clustering. However, we are pleased to see how, with almost 7,000 documents, we were still able to reach our goals.

Part-of-speech tagging

One of the advantages of analyzing text is that we have the possibility to use actions in the form of verbs, helping us to detect what we are interested in. In order to detect verbs, we use a part-of-speech (PoS) tagger, a function that assigns a grammar role to each word in a phrase. Since the process used to assign the tag is based on a set of heuristic rules, the procedure can be faulty, especially when new words are presented in input (this happens often with the name of certain drugs). However, for known and common words we were able to extract what we were looking for: the gerund of verbs to understand if an action was in progress or not ("the driver was calling" vs "the driver called").

Filter off-topic cases

To identify topics, we compute N-grams. Building a database containing all the N-gram information is very expensive from a computational point of view. We estimated that, in our case, this would have led to a structure containing over 10 million unique expressions. To reduce this number, we needed to find a way to filter some of the phrases created by the normalization procedure. One way to accomplish this task was to search phrases that contain words that are crucial to our application. Using the list of substances created with the word2vec model, we were able to filter the phrases, thereby reducing the database to a more manageable size. The filtering process is quite computer extensive. However, 97.6% of phrases are labeled as off-topic (see Table 4).

Table 4: Recap of the filtering process

	# Case	# Phrase
Filter	2,679	4,583
Off-topic	4,270	193,870
Total	6,949	198,480

Table 4 shows that 4,270 accident narratives can be labeled as off-topic since none of their phrases mentioned the presence of drugs or alcoholic beverages. Drugs or drinks are mentioned in 38.5% of the total narratives. We still cannot say if those factors are really present (or not present) because we could have stumbled upon a false positive, for example, cases that mentioned drugs or drinks only to state that they were *not* present. To overcome this pitfall, we implemented an N-gram strategy to understand text. Until now, we have focused on a unigram model that only considers whether specific terms are mentioned in the text.

N-Gram structure and flag creation

Building the N-gram structure is only the first step towards extracting information from the accident narratives. To be able to understand the expression written in the documents, we created a series of flags that exploit the structure we have created. These flags were then used to indicate the presence of drugs, illegal narcotics and alcohol in the accident narratives. Four different N-grams were investigated:

Unigram

We created four different flags:

- Flag 1 indicates the reference to medications in the text
- Flag 2 indicates the reference to prescriptions in the text
- Flag 3 indicates the reference to narcotics in the text
- Flag 4 indicates the reference to alcohol in the text

As described above, focusing only on single words does not allow us to get the full context. We are not able to catch negations and other more complex text expressions.

Bigram

Hence, we expanded our focus to pairs of words. Exploiting bigram, we tried to detect two particular patterns:

- negations: a flag is created to indicate the presence of a negation (no, non, not, none) followed by the name of a drug or an alcoholic beverage.
- actions: a flag is created to indicate the presence of an action (verb -ing) followed by the name of a drug or an alcoholic beverage. We are interested in the gerunds because they are commonly used to express real, fixed or completed actions.

Transitioning from words to pairs of words has many implications. On the one hand, we could refine our search for substances by accounting for particular combinations of terms. On the other hand, we needed to deal with the ceiling represented by the complexity of the analysis. In the previous step, we had to analyze only 3,261 unigrams, while 16,692 bigrams were generated.

To take an example, using the phrase of the accident narrative highlighted in bold in Figure 3 produces the following bigrams:

1. driver was 2.was crash 3.crash three 4.three weeks 5.weeks prior 6.prior crash 7.crash was 8.was taking, 9.*taking medications 10. medications ibuprofen 11.ibuprofen Vicodin 12.vicodin ultram*

Only a fraction of these bigrams is effectively used in the analysis.

Trigram

Regarding trigrams, there are 24,900 combinations of words triplets, 49% more than the size of the bigram structure and almost 7 times more with respect to the unigram structure.
Considering triplets instead of pairs of words further increases the complexity of the analysis since numerous expressions can be encoded in a larger window. Aside from the classical controls already implemented, we further refined our searching algorithm by taking into account:

- Past continuous, so that we now track every triplet which contains a reference to this verbal form. Typically, the past continuous is used to express past events that were prolonged in time, or that continued before and after other actions. A classic example is the triplet "was doing drugs". Notice that in this specific triplet a flag indicating the presence of an action was already implemented in the unigram phase. However, the bigram "doing drugs" does not allow us to exclude the fact there was a negation before the verb *was*. Therefore, to refine the search for enduring past actions we implemented this check, which also has the benefit of removing any doubts regarding negations or other contradictory expressions.
- Past tense: we enhanced the check regarding past tenses by tracking down triplets that contained the past declination of the auxiliary verbs *to be* and *to have*. This phase allowed us to consider expressions such as: "was on medication", "had taken percocet", etc.
- Negative and positive tests, so that we can track the presence of toxicology or BAC tests in the narrative descriptions of the accidents. We are only dealing with phrases that contain at least one occurrence of a substance, so it is fair to assume that any mention of the terms positive and negative is related to some sort of intoxication test. Exploiting this assumption, we proceed to detect triplets such as: "cocaine negative results", "negative results drug", "alcohol tested positive" and "positive opiates cocaine".

It is clear that triplets offer a broader view on the context of phrases. Just like a baby learns to read, the algorithm takes baby-steps toward a good comprehension of the meaning of documents.
As emphasized above, we are not declaring an omni-comprehensive list of all the possible expressions that can point toward the correct identification of substances during accidents, which would take too much time. We are only trying to characterize a small set of rules that are at least able to understand the general context in which the crashes happened.

Fourgram

The design of the last batch of checks implemented is based on the fourgrams and will be used to detect particular samples of false positive patterns. The degree of complexity increases again, from the previous 24,900 instances of the trigram to the 27,642 of the fourgram. We use this structure to detect the following patterns:

- Prescription medication; in the accident narrative it is common to come across this pattern. Reading this narrative, we know that it refers to a prescription; however, in the previous steps of the analysis we may have assigned contradictory flags to it. Because of the presence of the word medication, the algorithm may be deceptive and falsely attribute this pattern to the wrong class. By searching for this pair of terms in a broad context, we were able to contextualize the terms and correctly classify them in the prescription category.

- Prescription sunglasses, another pattern (as the previous one) that can be difficult to deal with. The presence or absence of prescription sunglasses (lenses or other eyewear items) is not one of our goals; therefore, we need to find a way not to include them in the analysis. In order to exclude these patterns, we can build a flag that marks them as "off-topic", so that they will have little importance in the overall picture.

Realistically, many patterns may escape our rule-based algorithms, but we are counting on the fact that the flags created in the unigram structure act as a safety net. This concept has its foundation from the filter phase executed upstream. Having retained all the phrases that present at least one occurrence of the presence of the substance, we are 100% sure that in the relative cases there is at least one substance involved. By trying to exclude from the picture all the possible negative expressions, we are only exposing ourselves to the possibility that some patterns for positive detection may be ignored. Despite this fact, by having structured flags that act only on the mere presence of substance names, we are hedging our exposure with regard to missed opportunities.

Finally, we have obtained a list of 24 text-related flags (e.g., "she/he was/was not taking medications/prescriptions/drugs", "she/he was positive/negative to alcohol test", "she/he was/was not wearing sunglasses", "she/he was/was not calling someone", etc.), where each of them is used to detect a particular pattern in the N-gram structure. Therefore, a flag (0/1 for the absence/presence of a particular pattern) will be associated with every phrase.

Prediction model

The flags can be used to investigate the effect of a pattern on accidents. For example, the evidence of substances (legal and illegal) and their impact on motor vehicle crashes can be analyzed by studying the percentage of injuries (both minor and more serious, deaths included): we expect the percentage to be higher than the one in accidents where these factors were not present. By using a logistic regression with an elastic net regularization (Friedman et al. 2010; Zou, Hastie, 2005) to perform regression and variable selection by linearly combining both L1 and L2 penalties, we have estimated the probability of injury in accidents in which at least one substance was present. A justification of the need of this methodology is that text mining problems can be treated as $p > n$ problems, where *p* are the predictors and *n* are the records. Moreover some variables may be highly correlated and methods such as *lasso* regression will tend to pick only one of them and ignore the others without caring to which one has been selected. This is a limit that we do not find in *ridge* regressions. If the variables are highly correlated when the task at end is described by a $n > p$ situation, it has been shown that the ridge regression always outperforms the lasso but it does not allow to filter unnecessary predictors. The limits of the previous models do not belong to the elastic net algorithm, a regression analysis method that combines both the lasso and the ridge regression overcoming their shortcomings. Shortly in our case consider the logistic model

$$\ln \frac{\Pr(Injury|\boldsymbol{x})}{1-\Pr(Injury|\boldsymbol{x})} = \beta_0 + \boldsymbol{x}^T\boldsymbol{\beta}$$

Let $y_i = 1$ if the *i*-th report returns at least one injury or $y_i = 0$ on the contrary case. Let $p(\boldsymbol{x_i}) = \Pr(y_i|\boldsymbol{x_i})$. Then the aim is to maximize the penalized log likelihood

$$\max_{\beta_0,\boldsymbol{\beta}} \left[\frac{1}{n}\sum_i \left(y_i \ln p(\boldsymbol{x_i}) + (1-y_i)\ln\big(1-p(\boldsymbol{x_i})\big) - \lambda P_\alpha(\boldsymbol{\beta})\right]\right.$$

where $P_\alpha(\boldsymbol{\beta}) = (1-\alpha)\|\boldsymbol{\beta}\|_{L_2}^2 + \alpha\|\boldsymbol{\beta}\|_{L_1}$, $0 \le \alpha \le 1$ and $\lambda \ge 0$.

Figure 5 shows for all the factors (flags) we considered how the probability of injuries ranges from 77% for alcohol to 83.4% for illegal narcotics[5] and that it is always greater than the benchmark computed by using only the off-topic cases (see also Borba, 2013, for similar results obtained by using logistic regression).

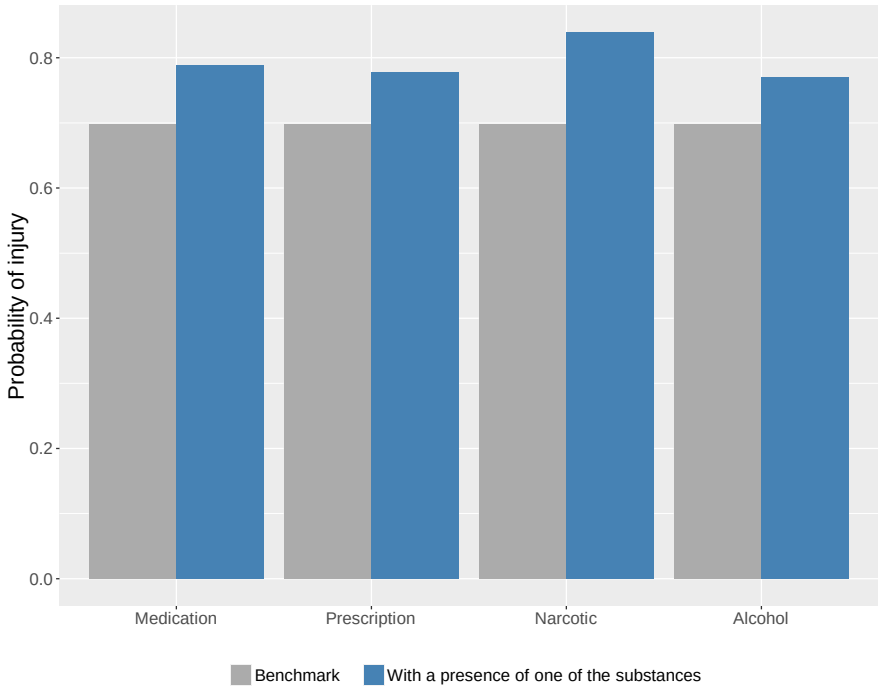

Figure 5: Probability of injury due to the presence of substances

All factors present results that comply with our initial guess. It is more likely that injuries will occur in crashes where at least one of the drivers had taken drugs or was under the effect of narcotics and alcohol. It is interesting to note that the probability of injury in the presence of medications (78.78%) is higher than the cases in which prescription medicines are involved (77.7%). It is hard to say if the two percentages are significantly different. Medication drugs are available over-the-counter, so

[5] We remind that off-topic cases are those reports where no medication, prescription and alcoholics were not mentioned (4,270 cases).

they are easier to access and should not affect common activities such as driving. However, it is possible that the abuse is higher because of how easily they can be bought. On the other hand, with medications we refer to antibiotic and other common drugs that are used to treat seasonal illnesses. Therefore, it is possible that in particular seasons more people are effected by allergies, or simply a cold, and in order to cure them they use medication drugs. This increases the number of individuals driving under the effect of medications, thus increasing the probability of injury, therefore biasing its real value. In general, this effect is difficult to measure because it is not available into the reports and it can be hardly measured. The results regarding alcoholic beverages and illegal narcotics are in line with expectations: it is well-known that drug-impaired driving is a crime and can have some serious consequences. The same holds for alcohol-impaired driving, but we need to make a distinction. In the case of illegal narcotics, it may be not easy to detect DWI (Driving While Intoxicated) cases just after a crash. Tests for narcotics must be done by lab tests which may take time. What can be recorded is the possession or possible use of narcotics. We need a list of substances and a series of flags that test their absence/presence at the time of the crash. For alcoholic beverages it gets trickier. Determining DUI (Driving Under the Influence) might be difficult because of the different laws on the matter (in the United States, it is possible to drive with a BAC of 0.08%, while for drivers under 21 the zero tolerance law imposes stricter limits, between 0.01% and 0.05%) and because sometimes in the description it is generically said that alcoholic beverages have been found in the car. This means that, even if a document mentions that a driver had been drinking, this does not mean that he was driving under the influence of alcohol. In our case, we do not aim at determining the impact of alcohol infractions (DUI) but the general influence this factor can have on the way we drive. The results speak clearly: the mere mention of alcoholic beverages in an accident description is sufficient to detect crashes in which the probability of injury was higher than the benchmark.

Final step

Insurers may consider this data useful in the process of claim evaluations for the following reasons:

1. Detecting the presence of substances has implications on multiple levels: claim triage, subrogation opportunities and decisions on whether to renew contracts.
2. Claims adjusters can exploit this information to investigate the circumstances in which crashes occurred, thereby obtaining a more reliable basis on which to begin settlement negotiations.
3. Actuaries can use this information to refine their pricing procedures. Typically, actuaries base their policy premium calculation on a so-called “Company DB”, a dataset containing details about their customers. Intuitively, a company cannot ignore what is going on outside of its pool of clients, and to price correctly new and existing products it needs to consider phenomena that affects the entire population of non-customers and customers.

By using both structured and unstructured data extracted from the accident narratives, we can build a so-called “Cloud DB”. This data source should have some classes in common with the Company DB: age, gender, ethnicity, license state, etc. We can interpret these variables as identifying a risk profile that is common to the sources of information. This allows actuaries to use this text-related information in the definition of premiums, opening the door for a more extensive use and exploration of documents in order to discover new risk covariates. We do not aim at detecting which

individuals are taking narcotics when writing a policy but to explain how this information can be eventually used in order to modify premium rating of policyholders. The rationale of this process is not new in ratemaking process. First, we compare the risk of those that have been involved with those that have not been involved into accidents given a set of covariates. Then we extend this risk to all the customers eventually mitigating the effect for those that were safe drivers, but not putting their risk to zero. The result is an integration into the premium model algorithm of a latent variable that represents the potential risk of DUID (driving under the influence of drugs).
On a more detailed level, our application could analyze the characteristics of each single driver involved in the accidents. Using the fields derived from the XML cases, we were able to build a database with information regarding 12,300 drivers, such as:

- license origin, status, endorsement and restrictions
- age , sex (see Figure 6), height, weight and illness
- external factors (emotions and stress)
- external factors (in a hurry, fast follow and traffic)
- external factors (distractions and conversation)
- season, day of the week and time of the accident
- year of production and maker of the vehicle
- emergency transport in ambulance
- severity

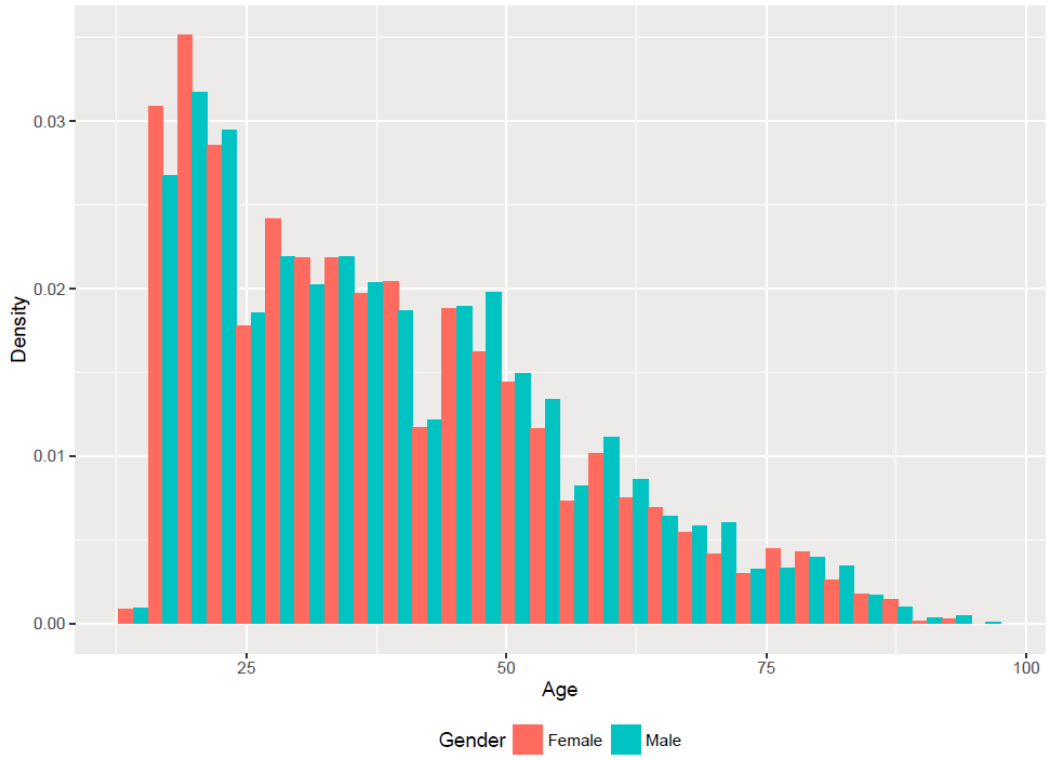

Figure 6: Distribution of injury by sex and age

We can match these profiles from the policy reports with the profiles of your own company database to add new risk covariates (i.e., probability of using substances or, for some categories, of being under the influence of external factors). To sum up these factors, in practice we built a model that classifies the accident outcome of each driver to predict if drivers were injured or not during motor vehicle accidents. The variables we used as predictors were those extracted from the XML cases in a structured form, including the four text-related flags referring to medications, prescriptions, illegal narcotics and alcohol. To test the classification accuracy for each driver we used an elastic net regularized regression.

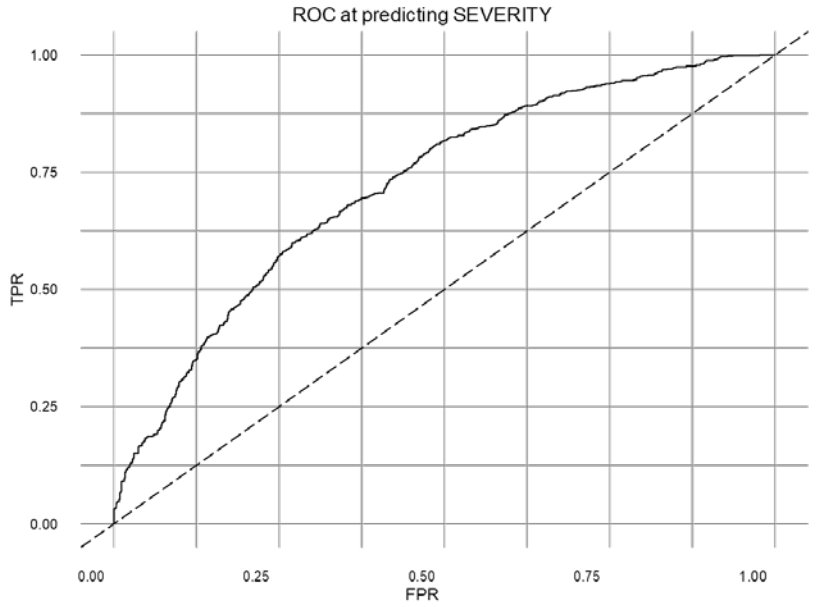

Figure 7: Receiver operating characteristic of the method

Figure 7 reports the ROC (Receiver Operating Characteristic) of our classification procedure, which can be interpreted as the ability of the method to correctly identify the classification categories. In our case, we obtained an area under the ROC of 72.25%, a good measure for a classifier that needs to take into account multiple variables and that uses predictors coming directly from the text.

Conclusions

We provided a case study that shows how valuable text mining can be for actuarial procedures in pricing models. However, the analysis could only scratch the surface of the possible applications that artificial intelligence methods may have in actuarial and insurance contexts in the next future. The aim was to create text-related risk covariates that may affect ratemaking.

The same process can be reproduced even using different contexts. New applications are emerging in terms of analyzing the GPS coordinates of social media posts to obtain proxy measures for the exposure to accidents. Researchers have found a way to use Twitter posts to estimate *drinking-while-tweeting* patterns to detect particular string patterns (*features*). This is another example of how it is possible to create new risk covariates linked to data mining of unstructured text documents.

The steps of the algorithm we propose are based on a data-driven process. In each step, the algorithm becomes "more conscious" of the context in which it is operating. It first detects the substances that are present in the documents and then proceeds to learn about the expressions that are present in the text. Each step allows us to create specific flags that increase the

understanding of the text. The algorithm exploits concepts coming from different areas, thereby avoiding the use of black-boxes.
Future challenges regard the applications of these methods to business activities using topic modelling. The insurance industry is well suited to the implementation of these advanced models because of the large quantity of data available to carriers. Text mining represents one instance of unconventional methods that could be used to refine the traditional approach of insurers to measure risk profile of their policyholders, thus opening the way for a wave of innovation. The correct application of these methods could provide the foundation for extending a Data Mining approach to Big Data projects.

Acknowledgements

This work has been sponsored by the Casualty Actuarial Society (CAS). The authors wish to thank the Editor and two anonymous referees for their careful reading and the suggestions that helped to improve the quality of the paper. The authors wish to give special thanks to David Core, Director of Professional Education and Research (CAS), and Jody Allen, Professional Education and Research Coordinator, for their support.
Finally, the opinions expressed in this paper are solely those of the authors. Their employers neither guarantee the accuracy nor the reliability of the contents provided herein nor take a position on them.

References

Bash, E., *The text mining handbook*, vol. 1. Cambridge University Press, 2015.

Borba, P.S., Predictive analytics, text mining, and drug-impaired driving in automobile accidents, available at: http://us.milliman.com/insight/pc/Predictive-analytics--text-mining--and-drug-impaired-driving-in-automobile-accidents/, 2013

Borba, P.S., Demonstrating the Value of Text Data for Predictive Modeling, available at: https://www.casact.org/community/affiliates/sccac/0515/Borba.pdf, 2015

Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., Bengio, S., Generating sentences from a continuous space, 2015, available online as *arXiv preprint arXiv:1511.06349*.

Bühlmann, P., Drineas, P., Kane, M., Dan der Laan, M., *Handbook of Big Data,* New York: CRC Press, 2016.

Ceron, A., Curini, L., Iacus, S.M., ISA scalable and accurate algorithm for sentiment analysis of social media content," Information Sciences, vol.367-368, pp.105-124, 2016.

Clark, A., Fox, C., Lappin, S., *The Handbook of Computational Linguistics and Natural Language Processing*. Wiley-Blackwell, 2010.

Feldman, R., Sanger, J., *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data.* New York, Cambridge University Press, 2016.

Friedman, J., Hastie, T., Tibshirani, R., Regularization Paths for Generalized Linear Models via Coordinate Descent., *Journal of Statistical Software*, vol. 33, pp. 1–22, 2010.

Guelman, L., Gradient boosting trees for auto insurance loss cost modeling and prediction. *Expert Syst. Appl.* 39(3), 3659-3667, 2012.

Lappin S., Fox C. eds., *The Handbook of Contemporary Semantic Theory*, 2nd Edition, Wiley-Blackwell, 2015.

Liu, B., *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2015.

Mailvaganam, H., Text Mining for Fraud Detection: Creating cost effective data mining solutions for fraud analysis, 2005, available at: http://www.dwreview.com/Data_mining/Effective_Text_Mining.html

Mikolov, T., Chen, K., Corrado, G., Dean J., Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations*, 1–12, 2013a.

Mikolov, T., Chen, K., Corrado, G., Dean J., Distributed Representations of Words and Phrases and their Compositionality. *Nips,* 1-9, 2013b.

NHTSA, "National Motor Vehicle Crash Causation Survey: Report to Congress", DOT HS 811 059, 2008 (available at https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059)

Pennington, J., Socher, R., Manning, C.D., GloVe: Global Vectors for Word Representation, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543, 2014.

Porter, M.F. , *An algorithm for suffix stripping*, Program, Vol. 14 3, pp.130-137, 1980.

Rong, X, word2vec Parameter Learning Explained, 2014, available online as a preprint on arXiv:1411.2738.

Stout, N., Analysis of narrative text fields in occupational injury data, in: Feyer, A.-M., Williamson, A. (eds.) Occupational Injury: Risk, Prevention and Intervention, pp. 15–20. Taylor & Francis, London, 1998

Turney, P.D., Pantel, P., From Frequency to Meaning: Vector Space Models of Semantics, *Journal of Artificial Intelligence Research,* 37, 141-188, 2010.

Wiley, J.F., *R Deep Learning Essentials*. Packt Publishing Ltd, 2016

Zou, H., Hastie, T., Regularization and Variable Selection via the Elastic Net, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67 (2). Wiley Online Library: 301–20, 2005.