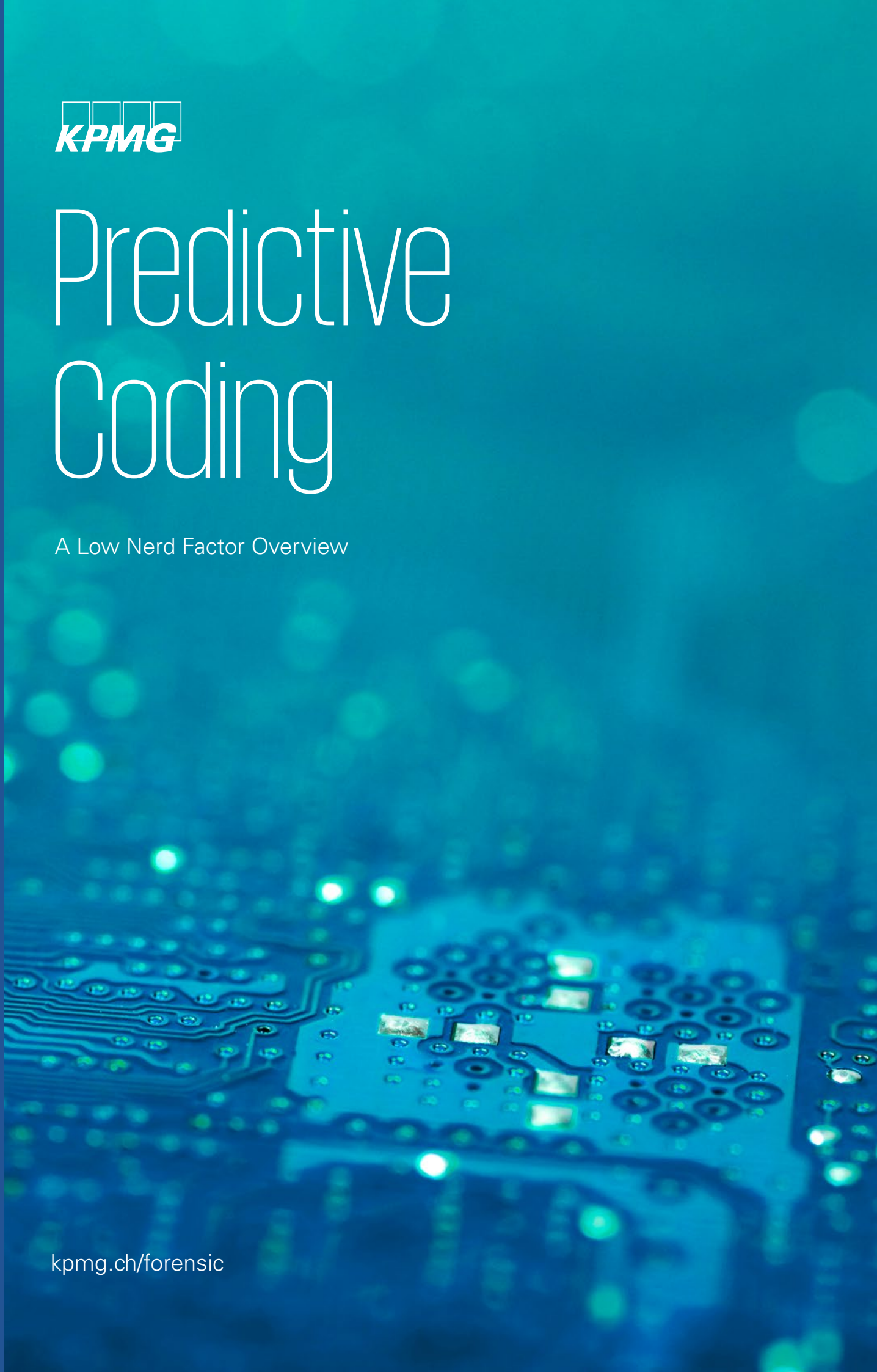
KPMG

Predictive Coding

A Low Nerd Factor Overview



kpmg.ch/forensic

Background and Utility

Predictive coding is a word we hear more and more often in the field of E-Discovery. The technology is said to increase the efficiency of E-Document reviews, decrease its cost and better the quality of the review. Many software provide predictive coding, we know Symantec® E-Discovery Platform, Nuix, Recommind® or Relativity® (1)

When doing an investigation on a large size organization (and nowadays even on a small one), the investigators often collect a large number of documents. In the computer era, creating, duplicating and modifying a document has never been that easy, hence a number of documents always increasing. Amongst this large number of documents, only a small fraction is of any use to the investigators. This small fraction needs to be retrieved and this is where predictive coding can help. It is used as a way to automatically classify a large set of Electronic Documents (E-Documents) usually into two different clusters (Relevant vs. Not-Relevant for instance) based on a small sample already classified by an experienced group of reviewers. The tool using predictive coding will learn from the sample what a relevant document looks like and identify the relevant documents in the set of documents not yet classified.

Predictive coding is studied in field of computer science called Machine Learning which comprises, among other things, artificial intelligence and data analytics, pattern matching (facial recognition, speech recognition), and signal processing. Predictive coding, also called automated classification, aims at building models of what data looks like and then compare the data to the model to see the resemblance. Many algorithms exist to build models, some are easy to understand, and other ones involve complicated math principles. But even the simplest ones involve probability. The goal of this article is not to explain on particular algorithm but rather to give a broad sense of what is going on behind the scene of such classifiers.

Contents

Tools and Math

It is not possible to talk about automated classification without talking about math. This part of the article is dedicated at making sure that word such as vector or n-dimensional spaces do not scare you.

A Bit about Languages

As said earlier, automated classifiers build mathematical models describing a document. Therefore, the first step would be to see a document as a mathematical object. In order to give a document a mathematical existence, we need to understand what an alphabet, a word, a document, a corpus and a dictionary are.

- Alphabet: it is set of symbols. For us the Latin alphabet is comprised of letters a to z to which we add capitals and accents. For a computer, the alphabet is binary 0 and 1. Each element of an alphabet is called a letter.
- Word: it is an ordered set of elements taken from an alphabet. For instance oeirut is a word build off of the Latin alphabet. Note that at this point languages do not exist yet. oeirut is not an English word but English has not been defined yet. The shortcut for a word is W.
- Document: it is an ordered set of words. In a document the same word can appear several times and each word has a place in the document. For a document D we will call Di the i-th word of the document, we will also define N(D, W) the number of occurrence of the word W in the document D. For instance, if D = "Predictive coding is not as simple as it seems" we have:

D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
Predictive	coding	is	not	as	simple	as	it	seems

And N(D, "Predictive") = 1, N(D, "as") = 2.

- Corpus: it is a set of documents. For example, the documents collected by an investigator from a corpus.
- Dictionary: it is the set of words that we can find in a corpus. For example, if we take the entire English literature as a corpus and we deduplicate the words we can find in that corpus, we will end up with the English dictionary.

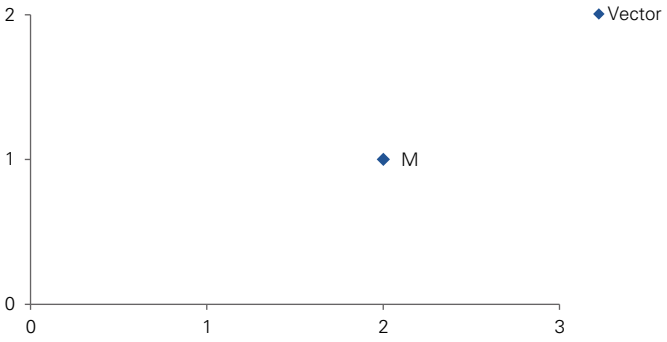
It is easy to apply these definitions to the languages that we speak. But it is to be noted that these definitions could also apply to many other kind of items. For instance, images could be seen as documents built upon letters that would be the colors and words that would be the lines of the image.

A Bit about Vectors

Now that we defined correctly a document, we have to see how we can bring them into mathematical objects. The operation will try to convert a document into a vector. Although a vector is a mathematical object that represents a document, it is not the document itself and we will see that there exist many ways to represent a document. Depending on the algorithms the documents will be represented differently, thus impacting performance and accurateness. But let's first explore what a vector is.

A vector is an ordered set of numbers. One such number is called a component of the vector and the number of components defines the dimension of the vector. We all remember the couple (x;y) that we spent hours to study in high school. Well, this is a vector of dimension 2 where the first component is x and the second is y. We also remember that we can write this vector on a plan: the point M below is described by the vector (2; 1)

Representation of a vector



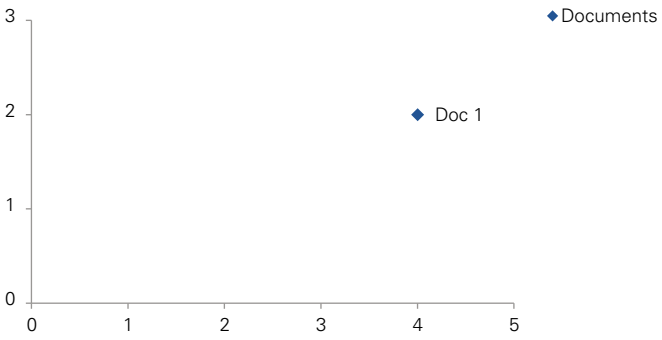
We often hear about two and three dimensional spaces (that we usually call plan and space). Sometime, we hear about four dimensional spaces (space to which we add the time dimension). In machine learning, we deal with spaces of up to millions of dimensions. Trying to visualize such a space would give anyone a headache.

Vectorizing a Document

We said that a document is an ordered set of words coming from a dictionary and that we need to transform a document into a vector in order to classify it. One approach often used is to create a vector of dimension n where n is equal to the number of words in the dictionary build off of the corpus of documents to classify. In English the number of words is roughly a quarter of a million (2). Then, the number of occurrence of a word W in a document is stored in the vector in the component accounting for W. Let's take a simple example, our dictionary will be comprised of the words a and b, those two words are composed of only one letter. The vector describing the documents based on these word will be of dimension 2 where the first component will count the number of occurrences of a and the second component will count the number of occurrences of b hence the document "a a a b b a" being represented by the vector (4; 2). Such a vector is called a feature vector as it describes the features of a document.

The graph below shows a representation of this document:

Representation of a document



There are at least three things to note:

- The order of the word is lost, the documents "a b" and "b a" are represented with the same vector (1; 1),
- The usual vectors are of extremely high dimension, the English dictionary is roughly 250'000 words meaning that the feature vectors representing a large set of documents written in English would be of dimension roughly 250'000, trying to display them would only lead us to a headache, hence this simple example,
- Representing the documents in such a way in a computer program would be highly inefficient, using few tricks allows programmers to achieve exactly this representation but in way more space-efficient manner. Usually if a word does not appear in a document, the corresponding component does not appear in the feature vector.

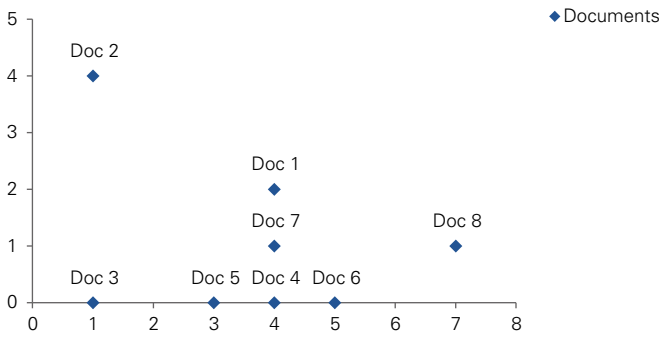
Classifying a Document

Now that we saw how to vectorize a document and to represent it in a visual way it is time to see how to classify a set of document. In order to achieve this goal, we will continue with our dictionary composed of the two words a and b and we will plot the following documents:

- Doc 1: "a a a b b a" represented by (4, 2)
- Doc 2: "a b b b b" represented by (1, 4)
- Doc 3: "a" represented by (1, 0)
- Doc 4: "a a a a" represented by (4, 0)
- Doc 5: "a a a" represented by (3, 0)
- Doc 6: "a a a a a" represented by (5, 0)
- Doc 7: "a a b a a" represented by (4, 1)
- Doc 8: "a a a b a a a a" represented by (7, 1)

Giving the following graphical representation:

Representation of a set of document

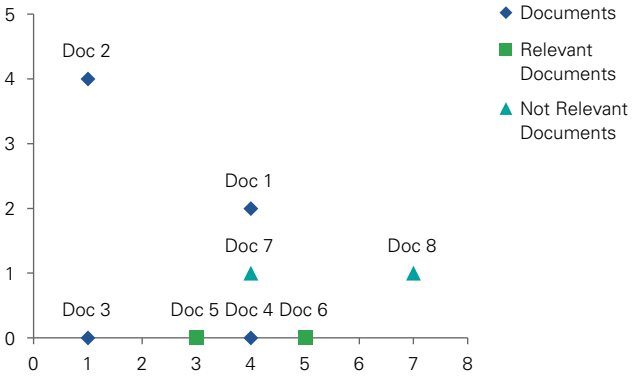


The idea now is to find way to group those documents into two different groups that we call classes. The simplest way to do this is to draw a line (or a plane in 3D) that separate the documents. Well that is exactly what a classifier achieve. The model is the definition of the line and if a document is above the line, it is said to belong to the class 1 (C1), otherwise it belongs to C2.

Note that in 2D, a simple line will separate the items, in 3D we will need a plane. In more generic terms, the simplest object dividing a space into two subspaces is a hyperplane.

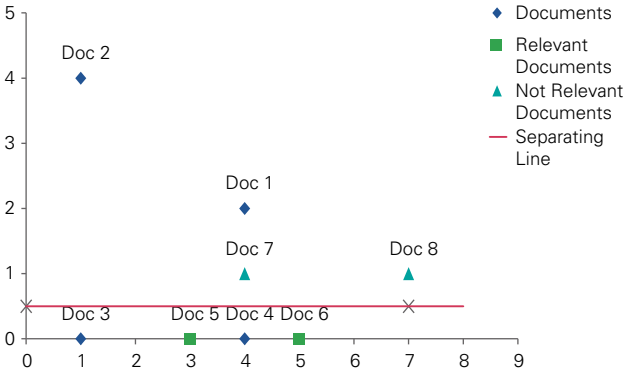
The classifier cannot invent a hyperplane, which is why it needs a training set from which it will deduct the parameters defining the hyperplane. Therefore, we need to tag some of the documents that are present as "not relevant" or as "relevant". This is the job of experienced reviewers and the result of their job is presented in the chart below where they tagged the documents 5, 6, 7 and 8:

Set of document with Training Set



A classifier could deduct from this training set that a horizontal line at the height of 0.5 would correctly classify the documents, it would mean that the document containing no b would be the relevant ones. The graph below show the line separating the documents tagged as "Relevant" from the ones tagged as "Not Relevant":

Set of document with Training Set



It is to be noted that the line separating the documents has been drawn in our example at a height of 0.5 but any line at a height above 0 and below 1 would work. A line that is not perfectly horizontal would also work. Given the training set, the choice of the line (hyperplane) depends on the classifier itself.

Summary

A classifier will transform a document into a feature vector; this vector will represent the document in a mathematical way such that it can be represented in a high-dimensional space. Then, an experienced reviewer will assign some of these documents (feature vectors) a class and from this assignation, the classifier will deduct a hyperplane that divides the space of documents into two partitions. The documents in one partition are said to be of class 1 the other ones of class 2. Many things can differ from a classifier to another; the following list is not exhaustive but shows several variants:

- The order of the word is usually not kept, but it is possible to give more weight to certain word. For instance a word that appear very rarely could see its weight increased while a word appearing in every documents could see its weight set to 0,
- The way the hyperplane is computed is the main source of difference between classifiers. It involves a lot of math, especially algebra and probability,
- The definition of what a word varies from one implementation to another (is "I'm" a word or two?),
- Many systems removes the word that carry no sense, such words are called stop words. Among them we find "a", "an", "the", "to"…They are useless as we find them in every documents,
- Some classifiers can add metadata to the feature vector, the idea is that the more information they have, the more precise they can be. But for practical reasons, it is not possible to add all the information at their disposal.
- The classifiers we described separate the space into two subspaces with a line, a plane, in general, a hyperplane. They are therefore called linear classifier. Non-linear classifiers theoretically exist, they would define a sphere for example and say that everything inside is C1 and everything outside is C2, but finding the parameters for a non-linear object requires much more computational power.
- As the considered classifiers try to separate the space into two sets, we end up with only two classes that are disjoint (no element in one class can be in the second class) and the classes are often described as C1 and not C1.

As final word, we will give two examples of classifier:

- Naïve-Bayes classifier (3) is a simple classifier often used to filter spam, It is also the one used in Nuix (4)
- Support Vector Machine (5) is a more complicated classifier used in Symantec® E-Discovery Platform (6)

Conclusion

In this article we explore how predictive coding work and gave a theoretical overview of the mechanism process that take place in such a tool. We explained that a document needs to be transformed into an abstract representation of the actual object. We then saw that this representation can be classified with the aid of automated programs therefore savings on time and staff.

The idea of this article is to give an overview of technology that we use or will be using in a near future. But, while those tools are more and more complex, only a short overview of the internal mechanism is sufficient to use them efficiently as they provide powerful user interface hiding this complexity. Our task is now twofold: be able to configure the tools correctly and make sure that the tools are performing as accurately as we want them to.

Bibliography

1. **ComplexDiscovery.** Predictive Coding One-Question Provider Implementation Survey – Initial Results. ComplexDiscovery. [Online] [Cited: September 10, 2014.] http://www.complexdiscovery.com/info/2013/03/05/running-results-predictive-coding-one-question-provider-implementation-survey/.

2. **Oxford English Dictionary.** How many words are there in the English language? Oxford Dictionary. [Online] [Cited: September 10, 2014.] http://www.oxforddictionaries.com/words/how-many-words-are-there-in-the-english-language.

3. Wikipedia. Naive Bayes classifier. [Online] Wikimedia fundation. [Cited: September 20, 2014.] http://en.wikipedia.org/wiki/Naive_Bayes_classifier.

4. **Nuix . Nuix and Predictive Coding.** [Online] Nuix. [Cited: September 10, 2014.] http://www.nuix.com/predictive-coding.

5. **Wikipedia.** Support vector machine. [Online] Wikimedia Fundation. [Cited: September 10, 2014.] http://en.wikipedia.org/wiki/Support_vector_machine.

6. **Symantec.** Clearwell 7.1.2 Feature Briefing. [Online] [Cited: September 10, 2014.] http://clientui-kb.symantec.com/resources/sites/BUSINESS/content/live/DOCUMENTATION/6000/DOC6731/en_US/Clearwell%207.1.2%20Feature%20Briefing%20-%20Transparent%20Predictive%20Coding.pdf.

This publication is a joint work prepared by Hedi Radhouane

Contact

KPMG AG
Badenerstrasse 172
PO Box
CH-8036 Zurich

kpmg.ch

Nico Van der Beken
Partner
Head of Forensic Technology

+41 58 249 75 76
nvanderbeken@kpmg.com

Anton Sieber
Director
Forensic Technology

+41 58 249 78 63
asieber@kpmg.com

Jakob Ogrodnik
Senior Manager
Forensic Technology

+41 58 249 79 68
jogrodnik@kpmg.com