TOWARDS SMART INVOICE PROCESSING: CRAFTSMAN LANGUAGE UNDERSTANDING

by Dr. Andrey Lutich

Digital processing and machine understanding of repair invoices is a serious challenge. The first step towards it is teaching machines to understand words similar to us, humans. For computers understanding means being able to represent words by numbers or vectors. State-of-the-art machine learning methods (e.g. GloVe, Word2Vec, fasttext) allow learning word vectors from unstructured text data. These vectors (aka word embeddings) allow machines to follow human intuition of language by clustering words by semantically similarity and enabling semantic analogy operations.

Invoice evaluation is a human prerogative

Alone in Germany in 2016 over 22 million of property damage claims have been processed by insurance companies. Naturally, the need to fulfil the damage claims resulted in handling and evaluating over 10 million property repair invoices or repair offers.

As of today, in Germany there is no standard nor a fixed format for repair invoice. Typical invoice is a wea-kly structured, human (craftsman) created document using a free-form natural language text to describe actions and materials required to carry the repair out. Furthermore, very often repair invoices do not con-tain explicitly all necessary information needed to assess them, rely on strong expert knowledge and the ability of in-depth understanding of invoice building blocks and their hidden interrelations. For example, if a repair invoice states that 5 m^2 of tiles were exchanged and 16 working hours were required, human experts would immediately realize that it is unusual and would question it.

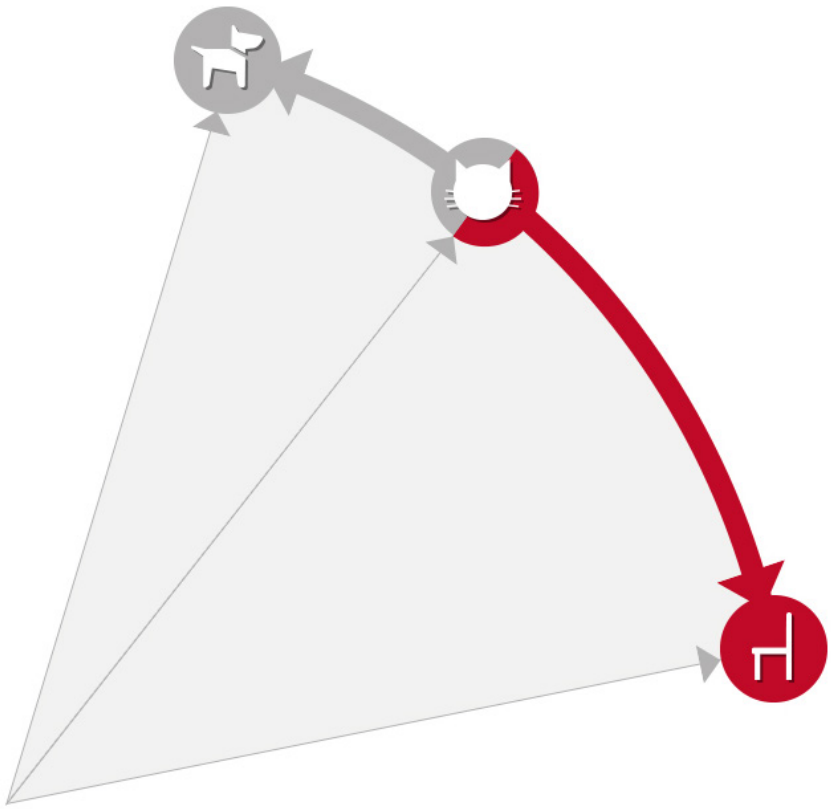
Making sense out of repair invoices today requires human-level intelligence and, therefore, is fully rely-ing on alive experts. The ability to automate this process offers obvious advantages for both insured and insurance companies: cost-efficiency, speed, errorless etc. The very first step towards teaching computers understanding invoices is to teach them dealing with natural language spoken by craftsmen in invoices and other repair-related documents.

Vector representation of words

Computers are designed to deal efficiently with numbers, not with words, in contrast to humans. Hence, to bridge this gap, words need to be converted to numeric format. The naïve approach is to count all the words used in a language and assign each word a number, for example its index. This method is simple to implement and it would allow computers to distinguish the words from each other, but it has a significant drawback: unrelated indices of words do not say anything about relationships between words. Discarding

connections between meanings of individual words is extremely inefficient because it requires computers to learn each word separately, without sharing information between words and their properties.

More natural and efficient approach to designing word representations (aka word embeddings) is mapping words to high-dimensional vectors and by design forcing these vectors to mimic commonalities between individual words. More precisely, while learning word embeddings, the distance between vectors representing similar words is minimized, and for words having nothing in common, it is maximized. For example, a human language carrier has clear feeling that the word “cat” is more similar to “dog” than to “chair”. This intuition is mimicked by having smaller distance between word embeddings of “cat” to “dog” than “cat” to “chair” as it is schematically illustrated in a 2-dimensional space. In real world applications, dimensionality of this space would be in the range from several hundred to several thousand.



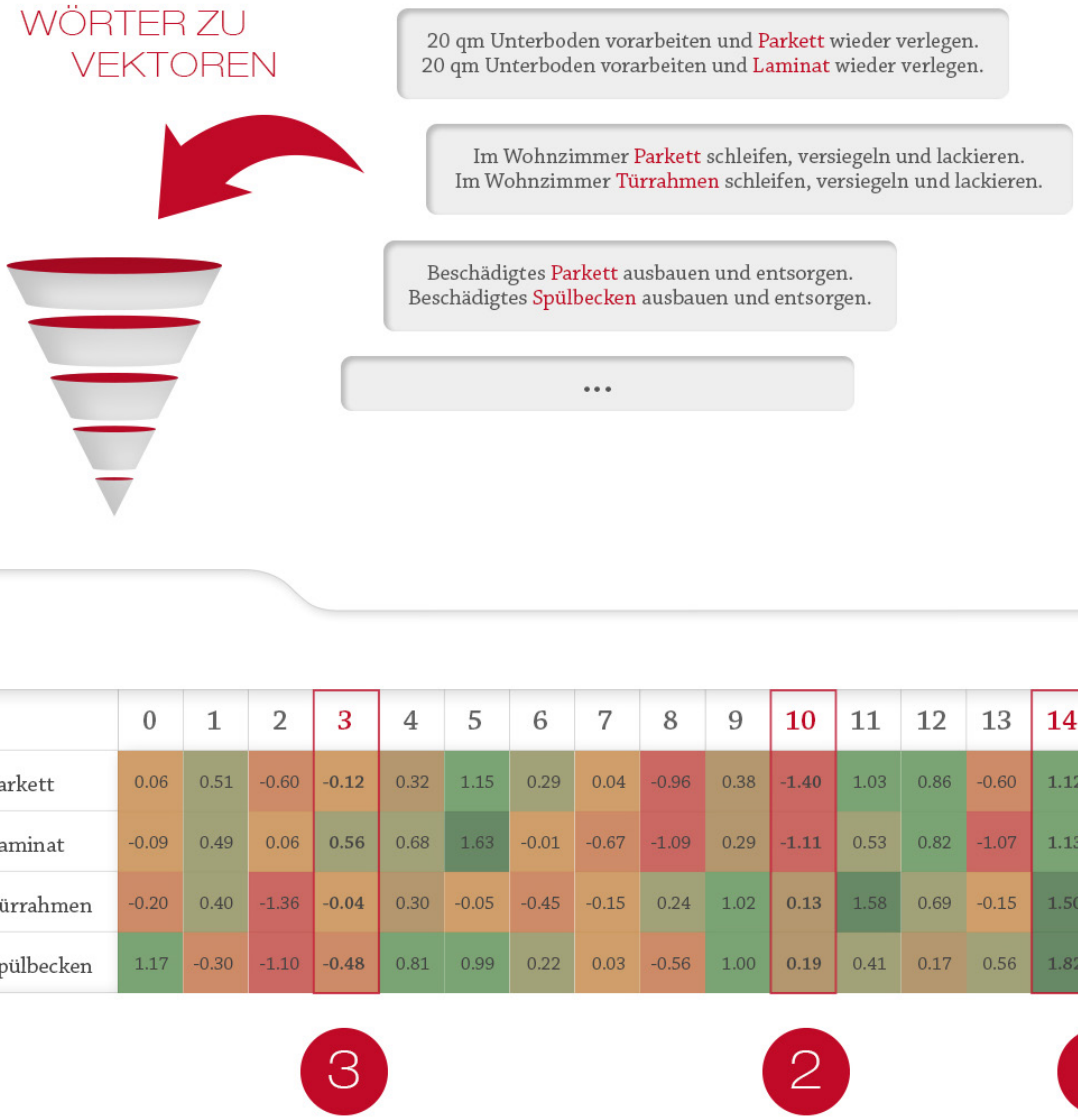
Once we decide to use such dense word representations, we need to have a way to assign each word such a vector. Certainly, performing this task manually and classifying all mutual word relationships in a language is not practical. Fortunately, the embeddings can be learned [1-3] from the way words are used in sentences written by human language carriers, in our case, craftsmen.

Learning word embeddings

Efficient method of learning representations of words is based on the observation that words having similar meaning are likely to appear in sentences written in natural language in the same or very similar context (surrounding words). For example, words that are synonyms can often be found used interchangeably within exactly the same sentence. In this way, meaning of words is learned by a computer not from words themselves, but from the variety of ways they are used in language constructs. This method is quite similar to the technique of learning foreign language by humans when, instead of providing word translation to a mother tongue, examples of word usage in the foreign language are given.

It is important to realize that objects or concepts described by words are very multi-faceted. Each word/object may possess a number of properties (color, shape, weight etc.) as well as numerous actions that are applicable to it (move, paint, lift, enjoy etc.) and other more complex attributes may be present. All these are different facets or aspects of words that are learned from contexts and encoded into multiple dimensions of word representations. Although, it is not always possible to assign specific meaning to a particular dimension of the embedding vector, general rule holds: the more similar representation vectors are, the more in common the words have. For example, the word “Laminat” has lot similarities to word “Parkett” and this is easy to understand. Nevertheless, it has also certain aspects similar to words “Türrahmen” and “Spülbecken” based on the fact they all can undergo the same repair operations and, therefore, often appear in similar context in invoice texts.

Various semantic properties of words are encoded by numeric values comprising the word vectors. For illustration purposes, word vectors are 16D (in real world applications word vectors have several hundred dimensions). It is not always possible and technically not necessary to assign human understandable meaning to individual components (numbers) of the vectors, but to get a flavor of the word vectors idea we can try it. Speculatively, we could state that vector components 14 and 15 (box 1) are describing some semantic properties that these four words have in common. Likely, the fact these are all parts of a house and they can undergo operations like damage, repair, exchange, insurance etc. Vector component 10 (box 2) is a property with respect to which [“Laminat” and “Parkett”] and [“Türrahmen” and “Spülbecken”] are similar within pairs, but very different between pairs. This could be the fact that “Laminat” and “Parkett” are both mounted on a floor in contrast to “Türrahmen” and “Spülbecken”, that are attached to a wall. The component number 3 (box 3), quite different across all words, may encode the material or methods used to fix described objects (e.g. glue, screws, nails etc.)



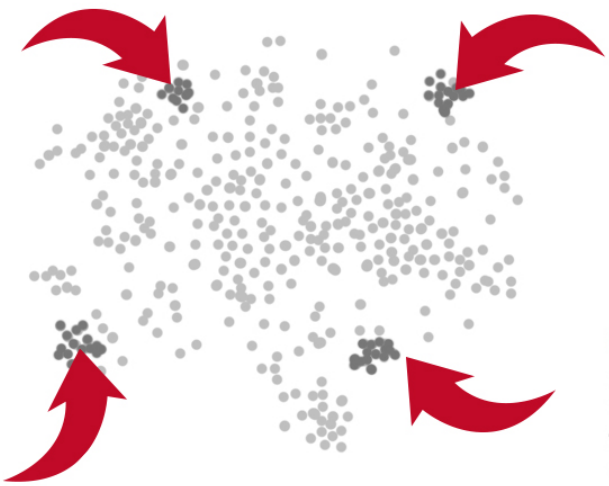
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Parkett	0.06	0.51	-0.60	-0.12	0.32	1.15	0.29	0.04	-0.96	0.38	-1.40	1.03	0.86	-0.60	1.12	-1.11
Laminat	-0.09	0.49	0.06	0.56	0.68	1.63	-0.01	-0.67	-1.09	0.29	-1.11	0.53	0.82	-1.07	1.13	-0.99
Türrahmen	-0.20	0.40	-1.36	-0.04	0.30	-0.05	-0.45	-0.15	0.24	1.02	0.13	1.58	0.69	-0.15	1.50	-1.15
Spülbecken	1.17	-0.30	-1.10	-0.48	0.81	0.99	0.22	0.03	-0.56	1.00	0.19	0.41	0.17	0.56	1.82	-1.19

3 2 1

The success of learning high quality word representations is almost fully determined by the quality and the amount of training data. Typical training corpus of text data, specific to a particular knowledge domain, consists of 10^7 – 10^{10} words. Ideal training corpora need to be as large as possible, but without redundancy and include all thinkable combinations of contexts in which individual words could be used in order to learn all fine details of interrelations between words.

Visualizing semantic relationships between words

Evaluating quality of language understanding/modelling is by definition rather subjective, as it is hard to define an objective measurable metric for it. To make sure that learned word representations make sense, relationships between words comprising the training corpus can be explored by a human and compared to human sense of language. We expect that words that are represented by similar vectors be also judged as having strong similarities by humans. Word vectors are hard to visualize directly because of their high dimensionality, but they still could be put into a 3- or 2-dimensional space using a dimensionality reduction technique, i.e. principal component analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE) [4].



Armatur	1.0
Unterputzarmatur	0.77
Mischbatterie	0.76
Wandarmatur	0.76
Armaturenoberteile	0.73
Badewannenarmatur	0.72
Küchenarmatur	0.72
...	

sachcontrol	1.0
sach|control	0.71
Bausachverständigen	0.62
Außenregulierer	0.55
Gewünscht	0.54
Schadenregulierer	0.54
Wertminderungsvereinbarung	0.53
...	

Müller	1.0
Schmidt	0.79
Hoffmann	0.73
Schiller	0.73
Schmid	0.72
Wagner	0.71
Reinhardt	0.70
...	

Bodenleger	1.0
Fußbodenleger	0.89
Parkettleger	0.72
Trockenbauer	0.70
Estrichleger	0.68
Schreiner	0.68
Raumausstatter	0.65
...	

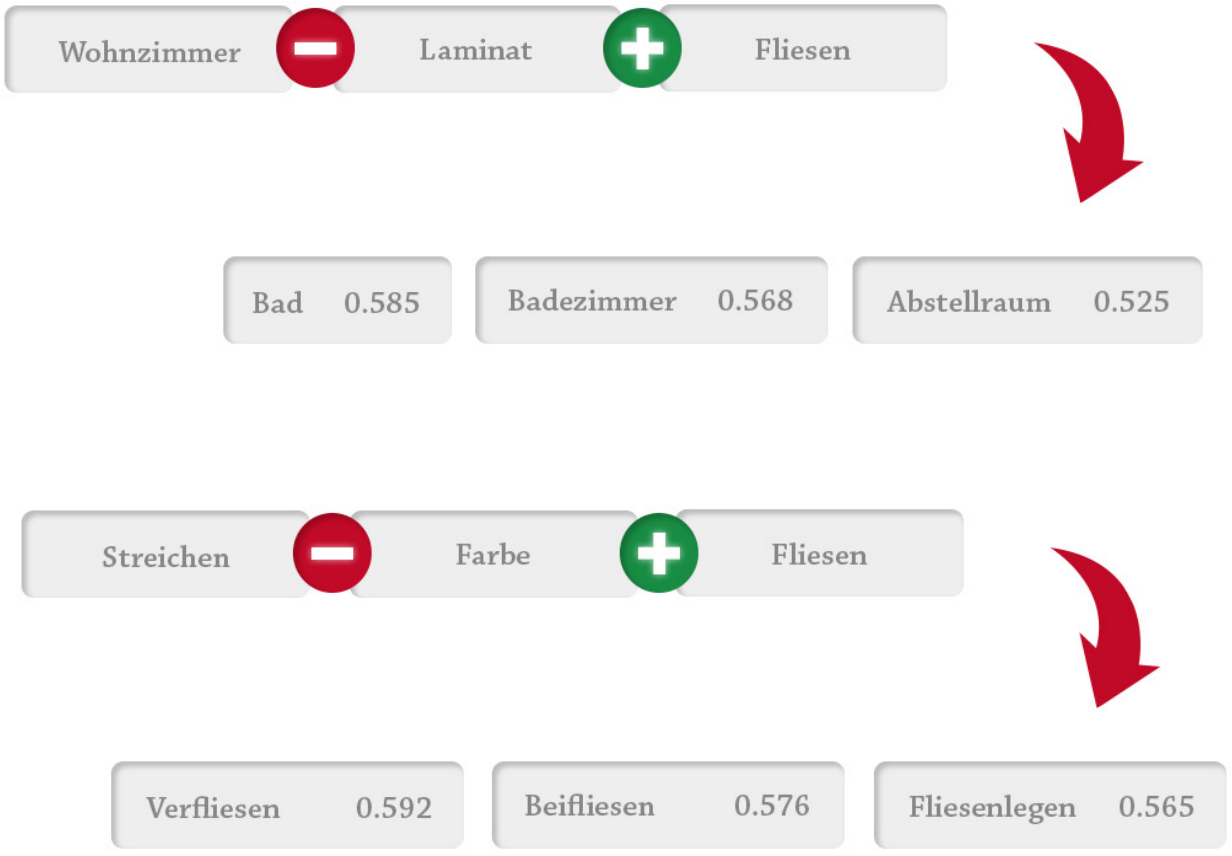
It is very exciting exercise to browse through the clusters of words and to discover surprisingly human logic behind these clusters. Here are just few examples. All types of “armatur” aggregate into one compact cluster highlighting the fact that our embedding algorithm learned to group these words together in spite of their completely different syntactic representations. The embeddings are also perfectly following human intuition by grouping craftsman specializations as well as typical German names into separate clusters. As the training corpus included numerous sachcontrol documents, the word “sachcontrol” had been seen by the model many times and, therefore, its’ vector representation has been learned as well. As one would expect, among others the services provided by the company and value-added deliverables are closest matches to the company name.

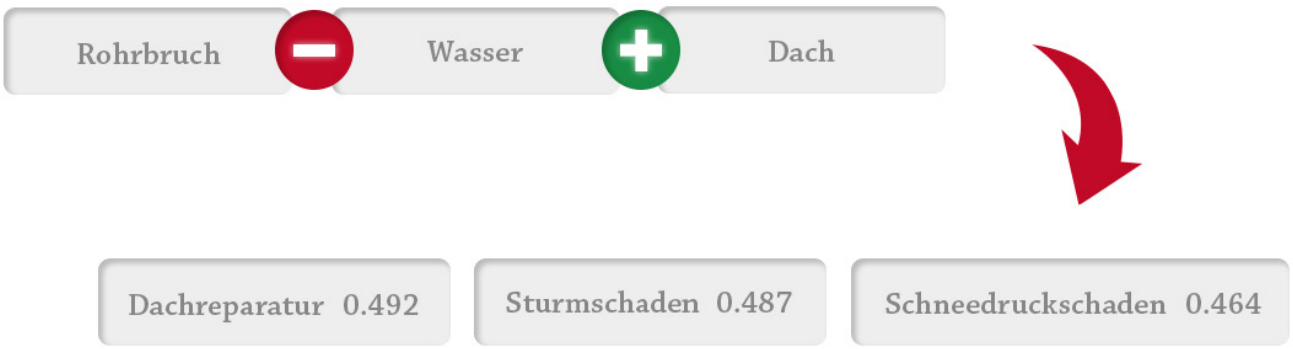
Math with words

Being able to represent words by vectors enables math directly on words, or more precisely, on the concepts behind the words. For example, using word vectors one could pose and answer semantic analogy questions, such as "Who is a King if he is not a man, but a woman?" This kind of questions can formally be expressed using arithmetic operations as "$Word_1 - Word_2 + Word_3 =?$" For the example question above correct equality would be "König – Mann + Frau = Königin".

To find answers to these questions first all three words on the left are converted to vectors, then arithmetic operations are applied to vectors and resulting vector is calculated. Few words from the vocabulary are selected that have highest cosine similarity to the resulting vector. These highest similarity words are the most probable answers to the posed questions. In the chart, three most probable answers together with their similarity are displayed.

The chart below illustrated how this semantic similarity math works with the repair related lexicon. It presents how the answers to the questions like "What is the room name if you take living room, remove laminate and put tiles?" or "What is that process of painting not with paint, but with tiles?" are derived.





Quite impressive, we are able to encode word meanings and extract essential information about words and their relations by applying simple linear algebra operation to their embeddings. The ability to perform this task has no immediate practical value, but it is an important test illustrating that our word embeddings have successfully learned fine details German craftsman lexicon.

Word vectors - done. What is next?

Words are the elementary building blocks of language constructs, sentences. Converting words to machine readable format, numeric vectors is the first step towards building human-level intelligence, capable of understanding complex structure of repair invoices. Word vectors, by design, carry rich semantic information. They are supposed to transfer this basic language knowledge into downstream machine learning algorithms that implement higher-level intelligence by learning it from human experts.

References:
1. https://nlp.stanford.edu/projects/glove/
2. https://www.tensorflow.org/tutorials/word2vec
3. https://fasttext.cc/
4. https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding