Lucas Woltmann, Maik Thiele, Wolfgang Lehner

Modeling Customers and Products with Word Embeddings from Receipt Data

Erstveröffentlichung in / First published in:

DEAS 2018: 22nd International Database Engineering & Applications Symposium, Villa San Giovanni 18.06. – 20.06.2018. ACM Digital Library, S. 246-252. ISBN 978-1-4503-6527-7

DOI: https://doi.org/10.1145/3216122.3229860

Modeling Customers and Products with Word Embeddings from Receipt Data

Lucas Woltmann, Maik Thiele, Wolfgang Lehner
Database Systems Group,
Technische Universität Dresden
Dresden, Germany
maik.thiele@tu-dresden.de

ABSTRACT

For many tasks in market research it is important to model customers and products as comparable instances. Usually, the integration of customers and products into one model is not trivial. In this paper, we will detail an approach for a combined vector space of customers and products based on word embeddings learned from receipt data. To highlight the strengths of this approach we propose four different applications: recommender systems, customer and product segmentation and purchase prediction. Experimental results on a real-world dataset with 200M order receipts for 2M customers show that our word embedding approach is promising and helps to improve the quality in these applications scenarios.

CCS CONCEPTS

• **Information systems** → **Information integration**; *Information retrieval*; *Enterprise applications*; *Business intelligence*; *Recommender systems*; *Clustering and classification*; • **Computing methodologies** → **Information extraction**;

KEYWORDS

word embedding, recommender systems, segmentation, applications, word2vec

ACM Reference Format:
Lucas Woltmann, Maik Thiele, Wolfgang Lehner . 2018. Modeling Customers and Products with Word Embeddings from Receipt Data. In *IDEAS 2018: 22nd International Database Engineering & Applications Symposium, June 18–20, 2018, Villa San Giovanni, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3216122.3229860

1 INTRODUCTION

With the emerging of the fully connected customer [13] in markets all over the world arises the problem of massive heterogeneous data collections for customers and products. Traditional methods usually fail to capture meaningful insights from this data []. One example are the collections of online receipts which contain merged information about both the customer and the products. In these data sets customer and products are not distinguishable as a single data point because an order item contains customer and product information. This makes it hard to compare customers and products and there impact on each other. We wanted to show the broad usability of Natural Language Processing (NLP) in a market research context to improve the quality of analyses from sources where customers and products are not modeled separately. We researched the possibilities of word embeddings and their application for modeling customers and products as comparable vectors. In this paper, we show that even just short product titles extracted from receipts are sufficient to build reliable word embeddings. Therefore, we extracted product titles receipts from online purchases to build a holistic vector space for words, products and customers. On top of that, we develop a product recommendation, a customer and product segmentation as well as purchase prediction system and evaluate their usefulness and quality.

We employed Word2vec [7] over a text containing all product titles in the receipts as paragraphs. This will give us vectors for each word. These word vectors will be aggregated to represent customers and products with vectors as well. Word2vec and its predecessor fastText [1], both well-known models to produce word embeddings, are powerful techniques to study the syntactic and semantic relations between words by representing them in a vector. By applying algebraic operations on these vectors semantic relationships such as word analogies, gender-inflections, or geographical relationships can be recovered easily. The probably best-known example is the "man is to woman as king is to queen" analogy which can be solved by looking for the word $\overrightarrow{w}$ such that $\overrightarrow{king} - \overrightarrow{w}$ is most similar to $\overrightarrow{man} - \overrightarrow{woman}$.

Recent papers already prove the usability of word embeddings to build recommendation systems, e.g. product recommendation systems based on emails [4] or item metadata [12]. Whereas these workings specialize on recommender systems, we used a broader approach covering more areas in market research.

The base for our analysis consists of 200M parsed customer receipts from online purchases from different suppliers and merchants from over 2M customers. As shown in Table 1, each order contains the purchased products with a certain price and the product titles which will be used to derive product representations in the following. This example shows very much the problem of non-comparable data points. The customer and the products share the same features in the data representation without actually sharing them in the real world.

Outline and Contributions The paper is organized as follows: in Section 2, we describe the derivation of word embeddings using fastText and their combination for customers and products vectors into one vector space. In Section 2, we present four systems using these word embeddings together with an experimental evaluation on real-world data. These four systems are

Final edited form was published in "IDEAS 2018: 22nd International Database Engineering & Applications Symposium. Villa San Giovanni 2018", S. 246-252, ISBN 978-1-4503-6527-7
https://doi.org/10.1145/3216122.3229860

user_id	product_id	order_id	order_total	product_title	product_price	quantity	...
eads-34	tef-34	09ed	28.98	Shampoo No. 5	18.99	1	...
eads-34	bhf-5	09ed	28.98	Adm. Ackbar's Cornflakes	9.99	1	...

Table 1: Data Example

(1) Purchase prediction,
(2) Customers segmentation,
(3) Product segmentation,
(4) Product recommendation.

Finally, we present related work in Section 4 and conclude this paper with a short summary and outlook to future work in Section 5.

2 WORD EMBEDDINGS FOR PRODUCTS AND CUSTOMERS

In this section, we want to introduce the models and methods used for embedding the words of the product titles in a vector space. Furthermore, we will describe how a combined vector space with word, product and customer vectors can be built.

For building a word vector space $\mathcal{V}$ from the product titles, we used fastText with a window size of 5 and a dimensionality of $l = 100$ which are the smallest recommended values for window size and l regarding [1]. These are not general rules, but rather rules of thumb. We tested our approaches on a vector space with $l = 300$ and did not find any improvements regarding their evaluation. With a value of the reduction to 100, calculations are faster and the memory footprints of our test set are smaller. We wanted to use the doc2vec capability of word2vec for reading products as documents, but by the date of publishing this feature has not been ported to fastText, yet. So, for generating a complete text from the product titles, each title is handled like a single sentence in a natural language. Each product title is used as a single line in a text document. The sum of all product titles therefore builds a continuous text which is necessary for making the textual input look like a text in a natural language. fastText needs this kind of input because of its heavy usage of context in natural languages.

As a second step, we need to lay the foundations for a comparison between customers and products, as the latter have very different sets of properties. Therefore, we employ a simple method for mapping the products to the vector space. For this, we calculate the centroid $\vec{p}$ over all terms $\vec{w}_i$ belonging to a product title $\mathcal{X} \subset \mathcal{V}$.

$$\vec{p} = \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|}\vec{w}_i \tag{1}$$

Since the resulting product vectors $\vec{p}$ have the same dimensions as the word vectors $\vec{w}_i$ they can be added to the vector space $\mathcal{V} = \mathcal{V} \cup C_P$ with $C_P = \bigcup \vec{p}$. Having derived the product vectors, we can also model the customers by calculating their centroid. This time, instead of using the words of a product title, we use the vectors of the products the customer has purchased. This changes Equation (1) to:

$$\vec{c} = \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|}\vec{p}_i \tag{2}$$

Here $\vec{p}_i \in \mathcal{P} \subset C_P$ and $\mathcal{P}$ denote the customer's purchased products. The *customer vectors* are also added to the vector space $\mathcal{V}$. It follows $\mathcal{V} = \mathcal{V} \cup C_C$ with $C_C = \bigcup \vec{c}$. The *combined vector space* (CVS) now consists of word, product, and customer vectors. Figure 2a illustrates the calculation of a product with three word in its title. Figure 2b shows a customer after purchasing three different products. This unified representation paves the way for comparable distance measurements such as the cosine similarity between the three types of vectors as well as algebraic operations that are needed for the following applications.

3 APPLICATIONS

In this section, we want to leverage the combined vector space (CVS) proposed in the last section to enable different recommendation, segmentation, and prediction tasks. We will present the general workings of each task, give examples on real-world use cases and evaluate their quality.

3.1 Purchase Prediction

Given the CVS with all its algebraic properties, we can use the centroid-based customer modeling approach for *purchase prediction*, where the goal is to find the product group of the next purchase of a customer. Usually, one utilizes the previous purchases for modeling the customer. Each purchase represents one product and is modeled by the respective product vector $\vec{p}$. To ensure that the purchase history of customer is large enough we focused on customers with at least 25 purchases. We model customers by calculating the centroid c of all their purchased products p_i with $0 < i < n-4$ using Equation (2), where n denotes the total number of all purchases of a customer. The last purchase (p_n in Figure 1) is used as the target label, describing the purchase we want to predict. The other four unregarded purchases, (p_{n-4} to (p_{n-1}, represent a gap to relax the evaluation measures in a later step. The division of purchases for an example customer is depicted in Figure 1.

Given the centroid c over the $n-4$ purchase of a customer the task is to predict the purchased product category p_n. By ignoring the time line of purchases we can solve this problem using a classifier, taking the customer centroids inputs and the purchased product category p_n as target label. We tested many different classification approaches but achieved the best results with a neural network having one hidden layer with 256 neurons. For optimization, the network utilizes a stochastic gradient descent to minimize the categorical cross entropy [9] [8]. To prevent overfitting, we added a drop-out layer with a 50% rate [11].

3.1.1 Evaluation. We have chosen different amounts of customers with at least 25 purchases each as shown in Table 2. In

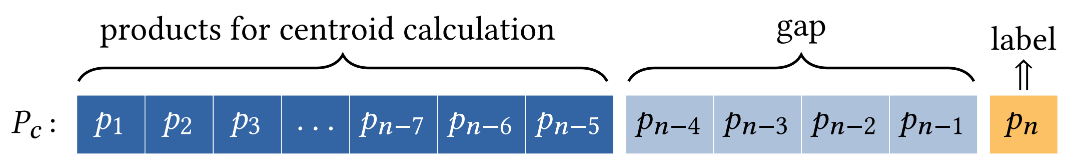
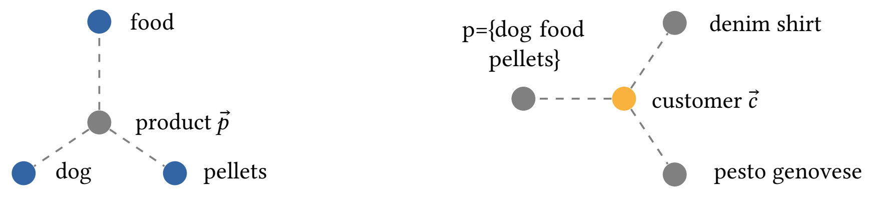


Figure 1: Purchase Data Model



(a) Product with product title words

(b) Customer with purchased products

Figure 2: Examples for centroid calculation in 2D

detail, our test setup includes four randomly sampled sets of customers. Each set is split into a training and a test set with 3-fold cross-validation. All test cases use 39 possible main product categories. After training our classifier, we evaluated its performance with three measures. These are *accuracy*, *accuracy@5*, and *accuracy in 5*. The *accuracy* is the ratio of correctly predicted product categories for the next purchase to the number of predictions to be made in total. *accuracy@5* is a subtype of precision@5 [2]. Accuracy@5 tries to predict whether the next purchase category is in the top 5 predictions of the classifier. Accuracy in 5 reverses this concept and tries to predict one of the next five purchase categories of the customer with the best prediction from the classifier. This is made possible by the gap we introduced earlier. Figure 3 gives an overview over both measures. Note that we still use the very last purchased product category as the target label during training. The purchases from fourth-to-last to second-to-last are ignored whilst training. This prevents the classifier from overfitting to the labels in the gap. For the test sets, we get an increase for the accuracy@5 and the accuracy in 5 of 20 to 40 percent points compared to the standard accuracies detailed in Table 2. The classifier can not give an exact prediction of a user's purchase, but it can still deliver some useful insights. For any customer we calculate two things: The accuracy@5 presents the next categories, where the probability of the customer purchasing is high. We can use this information to offer the user recommendations from these five categories. On the other hand, the accuracy in 5 will provide one category, from which the customer is likely to purchase in one of the next five purchases. This does not necessarily have to be the next purchase, but any of the next five.
The results show the feasibility of a purchase prediction using word embeddings as a data basis. To further improve the prediction accuracy one could use the status group classification and other rankings for products in the market.

3.2 Customer Segmentation

Advertisement and recommendation are usually chosen according to the customer or target group, which the customer belongs to. The general process of classifying customers into target groups is called *customer segmentation*. To derive a customer segmentation from the word embeddings presented in Section 2, we first need the notion of a *concept*. Specifically for the customer segmentation task a *concept* is a property of a customer that can be represented by the existence of one or several terms. The vector of such a term is called *concept vector*. In detail, we distinguish between contrary and unary concepts.

3.2.1 *Contrary Concepts.*

A concept that can be described by two opposing *concept vectors* $\vec{w}_{pos}$ and $\vec{w}_{neg}$ is called a *contrary concept*. The two words represent the semantic opposites of a status group. To classify a customer into target groups with contrary concepts, we calculated the cosine similarity to both concept vectors describing the contrary concept. The customer is then classified according to a positive or negative affiliation to the contrary concept by using the higher similarity. Equation (3) formalizes the idea with G_c as the statement about the affiliation, d_c as the cosine distance, $\vec{c}$ as the customer vector and $\vec{w}_{pos}$ and $\vec{w}_{neg}$ as the concept vectors of one contrary concept.

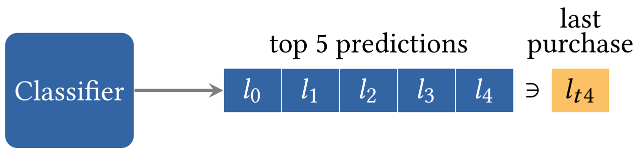
$$G_c = \begin{cases} 1 & \text{if } d_c(\vec{c}, \vec{w}_{pos}) > d_c(\vec{c}, \vec{w}_{neg}), \\ 0 & \text{else.} \end{cases} \tag{3}$$

Example: The contrary concept *marital status* can represented by the concept vector *marriage* and the opposite concept vector *alone* which includes the marital statuses single, divorced, and widowed. Similarly, the contrary concept and status group *house ownership* can be expressed with the concept vectors *house* and *rental*. These two words are at the ends of the semantic scale describing how people see home ownership.

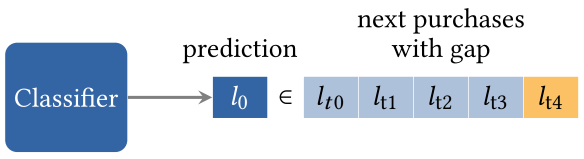
3.2.2 *Unary Concepts.*

Unary concepts have no alternative and therefore cannot be expressed with contradictory words, i.e. there are no matching words for not belonging to a unary concept. For example, owning a cat does not automatically mean that the customer owns a dog, whereas having no children automatically means that one is childless. To tackle this issue, we calculated the cosine similarity to just a single word $\vec{w}_{pos}$ representing the unary concept. We then normalized the similarities for one concept over all customers. The decision that if the user belongs to this concept is

Final edited form was published in "IDEAS 2018: 22nd International Database Engineering & Applications Symposium. Villa San Giovanni 2018", S. 246-252, ISBN 978-1-4503-6527-7
https://doi.org/10.1145/3216122.3229860



(a) Accuracy@5



(b) Accuracy in 5

Figure 3: Accuracy measures

training customers	test customers	number of purchases per customer	accuracy	accuracy@5	accuracy in 5
550	146	>25	53.2%	80.1%	84.2%
1600	400	>30	40.1%	81.2%	69.0%
20,000	4,000	>25	49.8%	78.1%	83.7%
43,000	10,000	>30	44.5%	75.7%	78.7%

Table 2: Accuracies for different sets of customers

correct if the normalized similarity to the word is greater than 0.4. The customer is assigned the label 1 for this concept. Otherwise the customer will be labeled with 0. Equation (4) formalizes the process with G_u as the label for a unary concept.

$$G_u = \begin{cases} 1 & \text{if } \left\| d_c(\vec{c}, \vec{w}_{pos}) \right\| > 0.4, \\ 0 & \text{else.} \end{cases} \tag{4}$$

We would like to point out that choosing an arbitrary threshold of 0.4 can lead to overfitting. An idea to overcome this issue is the usage of statistical analysis of the distribution of cosine similarities for a unary concept. We could use the average or other statistical measures over the similarities as a threshold. This would make the classifier less prone to overfitting.
Example: As a point of reference, we used the unary concept *cat*. Equation (4) assigns the label 1 for the unary concept to each customer if the given threshold is exceeded. To get a global view onto the unary concept, we can sum up all customers with label 1 and determine the ratio of customers that potentially own a cat.

3.2.3 Evaluation. To demonstrate the effectiveness of our idea to leverage word embeddings for customer segmentation, we choose three contrary and two unary concepts depicted in Figure 4. To compare the accuracy of our approach we got the actual segmentation of the customers into five status groups from a third party source providing us with a ground truth (see the left bars in Figure 4), in a privacy compliant way without revealing personally identifiable information. From the 2M customers in our dataset (see Section 2) we sampled 120,000 customers with more than two purchases and determined their segmentation using Equation (3) Equation (4). The results are shown in Figure 4. With an absolute deviation of 1 to 2 percent points and a relative deviation of 4 to 10 percent, we can show that the algorithm gets marginally close to the original ground truth. This means that customers can be modeled into social groups just by looking at their purchases. To be more specific, the language behind the product titles provides us with an understanding of how close a customer is to a social construct, like marriage. This assumption is quite clear for two reasons: The first one being the fact that the product titles mask several concepts by co-occurring words. The words *white dress* seem to be used more often for wedding dresses and are therefore closely aligned with the concept of marriage. The second reason is the origination of product titles. Products are always labeled for representing the maximum amount of information in as few words as possible. This makes product titles semantically dense. Thereby a product manager will give the product a name suitable for the desired target group. This aids our algorithm to find not only obvious concepts, but also words and products, which are not connected to the concept at a first glance. Note that the algorithm only works on a global level. For detailed information on a single customer one would need to find a better ground truth. For example by letting market experts label a subset of customer. Another critical point is the problem of shared accounts. If two or more people use one account for shopping and they belong to two different social groups, like male and female, one cannot make any assumptions about their social background. The purchases in this particular account would be too divergent and would subtract each other in the customer's centroid calculation.
There are still possibilities for improvement. A first step would be a more flexible threshold for unary concepts, like stated before. Additionally the concept vectors $\vec{w}_{pos}$ and $\vec{w}_{neg}$ could be expanded to include multiple words. Therefore, we could apply centroid calculation to average two or more words to represent one side of a contrary concept, e.g. the concept of not being married can be described with the centroid of the word vectors *single*, *divorced* and *widowed* instead of the word vector for *alone*. This assumption can also be reversed. The classification can model more than five concepts. One can easily employ the same algorithms to an additional contrary concept of *divorced* and *married* and classify customers accordingly. This would require a better test set where the labels

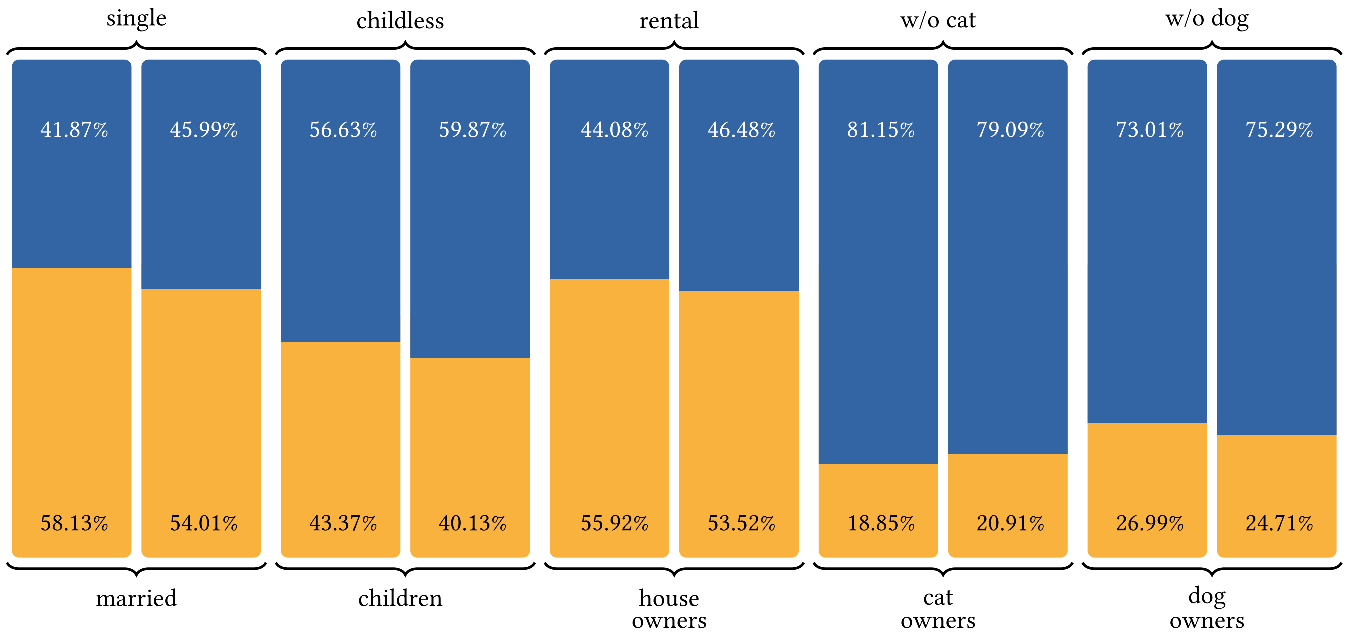


Figure 4: Original segmentation (left bar) and predicted segmentation (right bar)

for the concepts are available for every single customer and not on a global scale.

3.3 Product Segmentation

The concept of word representation for segmentation also works for products. We can use the concepts vectors on products in the same way as we did on the customers because of the combined vector space. Equation (3) can be rewritten to find the desired segment G_p of a product $\vec{p}$.

$$G_p = \underset{\vec{w}}{\operatorname{argmax}}\, d_c(\vec{p}, \vec{w}) \quad \text{with } \vec{w} \in \{\vec{w}_{pos}, \vec{w}_{neg}\} \tag{5}$$

The algorithm stays the same otherwise.

3.3.1 Evaluation. We tested the effectiveness of the product segmentation using word embeddings with one contrary group representing gender orientation of a product, i.e. *women* and *men*. Since we do not have any information about the gender orientation of a product, we crawled Amazon's apparel category for 310 product titles with gender labels. We have chosen a baseline classifier, which looks directly for the words *women* and *men* in the titles and classifies accordingly. For even better comparability, we also used an SVM trained on the tf-idf vectors modeled from the product titles.
We calculate the cosine similarities between products plus their respective titles and the two words *men* and *women* and label the product with the concept which has a higher similarity. In a second experiment we removed the words *women* and *men* from the product titles and rerun the segmentation. The removal prevents the classifiers from overfitting to the two concept words. Table 3 details all outcomes. One can clearly see the good performance of our product segmentation, even without the words *women* and *men* in the title. This points towards a stable classifier capable of tracking gender in any product title. We would like to point out that the Amazon product titles are not in the training set for fastText. It also reinforces the assumptions made about the stability of finding status groups. Product titles have a strong tendency towards concepts described by single words. The prerequisite still is a good model for the vector representation of the language in use. On the other hand, the language does not need to be a natural language. An artificial language, like the one for product titles, is enough to model a vector space.

3.4 Product Recommendation

The main goal of *recommendation* is to find products, which are similar to ones purchased before. Given the combined vector space we choose a customer as starting point. We then apply a *nearest neighbors approach* to get all products close to the starting point. To find the most similar neighbors of a customer, we use the well-known cosine similarity between each product and the customer. Items already purchased by the customer were excluded.

$$\vec{p_o} = \underset{\vec{p}}{\operatorname{argmax}}\, d_c(\vec{p}, \vec{c}) \quad \text{with } \vec{p} \notin \text{purchases of } \vec{c} \tag{6}$$

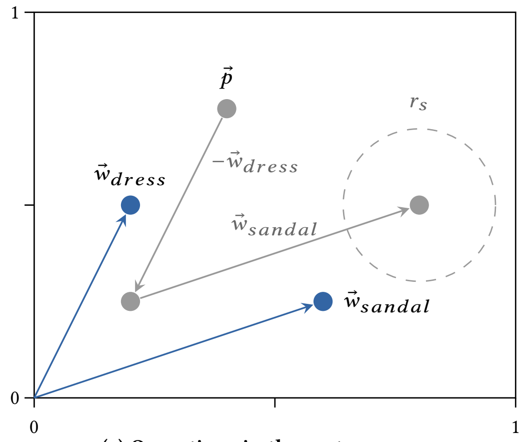
By applying this approach we might get false positives through the high variance in products close to a customer. These are recommendations of products, the customer is not interested in. The quality of such a recommender system is hard to access without the verification of experts. So far we do not have a gold standard for the recommendations. We only can access the quality of it by sampling some test customers. We have presented one of them in Table 4.
Our second approach utilizes the algebra operations that are possi-

[2]https://anon.to/raX6rU, [3]https://anon.to/SNjifY, [4]https://anon.to/fu7Vuj, [5]https://anon.to/zJ5XSa, [6]https://anon.to/5raj7U, [7]https://anon.to/yMyKA4

Final edited form was published in "IDEAS 2018: 22nd International Database Engineering & Applications Symposium. Villa San Giovanni 2018", S. 246-252, ISBN 978-1-4503-6527-7
https://doi.org/10.1145/3216122.3229860

product titles	baseline algorithm	SVM w/ tf-idf vectors	concept vectors
w/ *women* and *men*	88.7%	96.8%	97.7%
w/out *women* and *men*	0.0%	93.0%	93.5%

Table 3: Accuracy product segmentation



(a) Operations in the vector space

	h&m peplum dress ⇓
1	guess women's kymma strappy platform dress sandals
2	carter's strappy sandals
3	bcbgeneration ronny chunky heel sandal
4	b.o.c emmeline dress sandals
5	emporio armani black leather espadrille wedge heel t-strap sandals

(b) Top 5 recommendations

Figure 5: Product recommendations for finding matching sandals for a dress

marriage: tips and advice on how to maintain a healthy relationship [...]
self-help box set: techniques & strategies for greater mind power [...]
learn how to use focus to improve your concentration [...]
the art of becoming clutter free: decluttering your life in minutes [...]
time management [...]

(a) Book purchases

help guides to help you improve your skills and make your life easier [...]
amazing tips on how to improve your emotional maturity [...]
50 tricks to live a happier and successful life [...]
33 techniques to unlock the power of your mind [...]
the simple approach to health: a practical guide to understand your body [...]

(b) Book recommendations

Table 4: Book recommendations with nearest neighbors approach for an example customer

ble on the combined vector space. We revive the concept vectors from the previous chapters by representing product groups with single words. For example, the product category of dresses is summarized with the word vector for *dress*. Through the properties of fastText, all dresses are surrounding this word. They make up a cluster. This time, instead of a customer as starting point, we use a product $\vec{p_i}$ the customer c already bought as an input and want to recommend a product p_o belonging to a different product group, i.e. the product groups of the two products must be distinct $\vec{p_o} \neq \vec{p_i}$. Therefore, we take the product vector $\vec{p_i}$ as an input, subtract the word vector for the original concept of the item and add the word vector for the targeted concept of items. The result is a vector around which we employ a nearest neighbor search (cosine similarity) with a threshold r_s. The nearest neighbors are returned as recommendations. The result vector $\vec{p_o}$ is obtained from the word vector for the original concept $\vec{w_i}$ and the word vector for the targeted concept $\vec{w_o}$ via Equation (7).

$$\vec{p_o} = \vec{p_i} - \vec{w_i} + \vec{w_o} \tag{7}$$

Example: Imagine the customer bought a summer dress and is now looking for a matching pair of sandals. Figure 5a illustrates the calculation in the vector space.
First, we subtract the concept of dress from the product. Then, we subtract the vector for the word *dress* $\vec{w}_{dress} = \vec{w}_0$ from the product vector $\vec{p}$. After that, the vector of the target product group is added. This is done by adding the vector for the word *sandal* $\vec{w}_{sandal} = \vec{w}_t$. The resulting vector is the starting point for a search of nearest neighbors in r_s. Figure 5a also illustrates the proximities between products and concepts. One can clearly see how close the original summer dress is to the concept *dress* and how the derived vector is close to the concept of *sandal*. This means that the suggested items, found by the nearest neighbor search, are sandals. This give

us the results shown in Table 5b.It is clearly visible that all returned products are indeed sandals. A quick lookup confirms that the sandals match the style of the dress. Therefore, the algorithm can deliver recommendations which preserve stylistic information just by analyzing the language of the titles. This can be helpful in a recommendation context.
A recommender system could not only suggest products according to stylistic properties, but can also be used in an exploratory way. The customer can add concepts, like colors, which can be translated to a single word and the recommendation would prefer products with this color. For example, the user only wants sandals in black, so the algorithm adds the vector $\vec{w}_{black}$ to the final result from the algorithm and returns the products in the neighborhood of this new point. Note that this only works if the color can be found in the titles. Once again, we do not have a gold standard for the recommendations, so we had to access the quality by sampling customers.

4 RELATED WORK

The possibilities for modeling words as vectors is growing in numbers. The main difference between new algorithms and models is their capability of preserving semantic information. Simple models, like tf-idf [10], are easy to calculate, but lack encapsulation of semantic similarity or representation of concepts. On the other hand, more complex models like word2vec [7] or fastText [1] model these two aspects better, but their calculations take more time and need more data for building a stable model.
Neural network-based approaches are often used in recommendation, because they can be represent different concepts expressed by language. This is helpful when it comes to model review analysis [6] or sentiment analysis [3]. NLP and word embeddings have also been introduced to the field of market research [4][12]. Product recommendation can be enhanced by using language information delivered by products and their purchases. Another field for recommendation based on language usage are social networks. Social media content can be used to for recommendation by making use of the user's diction [5]. Usually, other approaches for product sentiment analysis use large amount of texts, like product reviews [14]. With our approach, we have shown that only using short titles for language modeling is enough to build reliable models.

5 CONCLUSION

This paper details improvements for product recommender systems and segmentation. One special improvement is the usage of very short text samples, i.e. product titles, compared to full product reviews [14]. Whereas product titles lack some vital properties compared to a natural language, like predicates or subject-verb-object-order they are still capable of producing a stable word vector space for recommendation and segmentation. Future research should focus on combining the segmentation and recommendation models to further improve their performance. We think including our recommendation and segmentation approaches into already existing systems as a hybrid system could build a more reliable source of information.

ACKNOWLEDGMENTS

The authors would like to thank GfK SE, especially Thomas Kirsche, Susanne Heckmann and Michael Schlueter, for their cooperation and help.

REFERENCES

[1] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.
[2] C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In *ACM SIGIR conference on Research and development in information retrieval*, pages 33–40. ACM, 2000.
[3] C. N. Dos Santos and M. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78, 2014.
[4] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1809–1818, New York, NY, USA, 2015. ACM.
[5] D. F. Gurini, F. Gasparetti, A. Micarelli, and G. Sansonetti. A sentiment-based approach to twitter user recommendation. In *RSWeb@RecSys*, 2013.
[6] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, pages 165–172. ACM, 2013.
[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.
[8] K. P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.
[9] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning.* Springer Science & Business Media, 2013.
[10] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24:513–523, 1988.
[11] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
[12] F. Vasile, E. Smirnova, and A. Conneau. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 225–232, New York, NY, USA, 2016. ACM.
[13] S. Wuyts, M. G. Dekimpe, E. Gijsbrechts, and R. Pieters. *The Connected Customer: The Changing Nature of Consumer and Business Markets.* Routledge, Taylor and Francis, 2009.
[14] J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. Sentiment analyzer: extracting sentiments about a given topic using natural language processing techniques. In *IEEE International Conference on Data Mining*, pages 427–434, Nov 2003.