# Identifying Customer Needs from User-Generated Content

**Artem Timoshenko (Northwestern), John R. Hauser (MIT)**
**Marketing Science, 2019**

Presenter: Shuyi Zhang

May 14, 2025

THE UNIVERSITY OF CHICAGO

# Introduction

Motivation:

- Traditional Voice-of-Customer (VOC) methods are time-consuming and costly.
  - Identification: Qualitative Interviews
  - Structuring: Manual reviews & summarize by multiple analysts.
- User-Generated Content (UGC) is an underutilized resource for product development.
  - Abundant unstructured textual data e.g. reviews, speeches
- Machine learning (ML) may offer efficiency gains in extracting valuable customer insights.

General Research Question: Can we utilize ML to effectively identify **customer needs** from UGC? If so, how well does it perform?

# Model: Summary of the System Architecture

**Figure 1.** System Architecture for Identifying Customer Needs from UGC

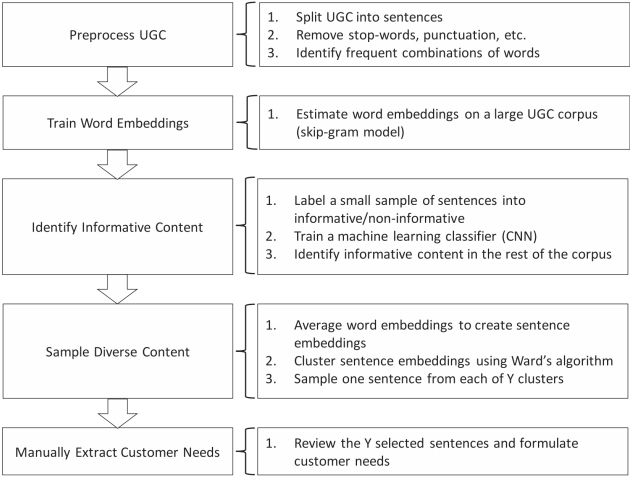



Figure: Timoshenko and Hauser (2019) Figure 1

# Data

- 115,099 oral-care reviews on Amazon spanning the period from 1996 to 2014, randomly sampled 12,000 sentences split into an initial set of **8,000 sentences** and a second set of 4,000 sentences.
- For the 8000 sentences, hired professional marketing analysts to fully **code every sentence to determine whether it contained a customer need** and, if so, whether the customer need could be mapped to a customer need identified by the VOC, or whether the customer need was a newly identified customer need.

To summarize, the data I focus on for the replication are:

- all_sentences: Al the oral care reviews.
- 8000_sentences: Fully labeled sentences. Informative = 1

# Data: Examples

**0**-*First of all, Target.com is selling the wrong unit.*
**1**-*:-)I'm achieving much better results with the Sonicare than with manual brushing, for 2 reasons: First, there's no way I come close to 31,000 brush strokes per minute by hand.*
**1**-*It's constantly getting dirty with dust and tooth paste.*
**0**-*I finally got this toothbrush after I have seen alot of people use them.*

# Model: 1. Preprocessing UGC

Steps:

1. Unsupervised tokenizer (nltk&regex) to eliminate stop-words(e.g. the, and) and symbols
2. Join 'frequently together' words into Phrases (e.g. even_though)
3. "We drop sentences that are less than four words or longer than 14 words after preprocessing."
4. Train Word Embeddings with **Skip-Gram Model**

# Model: 1. Preprocess UGC

Let's look at the following example[1] from *8000_sentences.csv*:

> *Well I really don't know how well the 4-pack of Teledyne piks (BRJ4) works because, even though it's clearly what I ordered, Goodman's, without notice, sent me the Official WaterPik 2-Pack (JT-70E) in its place for the same total cost to me as what I had paid for the Teledyne 4-pack.Two piks in place of four?.*

After the cleaning and tokenizing, we have:

- `['well', 'really', 'know', 'well', 'pack', 'teledyne', 'piks', 'brj', 'works', 'even_though', 'clearly', 'ordered', 'goodman', 'without', 'notice', 'sent', 'official', 'waterpik', 'pack', 'jt', 'e', 'place', 'total', 'cost', 'paid', 'teledyne', 'pack', 'two', 'piks', 'place', 'four']`

which, unfortunately, is longer than 14 words and is dropped.

[1] The outputs are given by demo_Preprocessing part in ipynb file

# Model: 2. Word Embedding Skip-Gram Model

$X$: A token in a specific sentence.
$E(Y)$: The expected probability of another within the 'window' of $X$

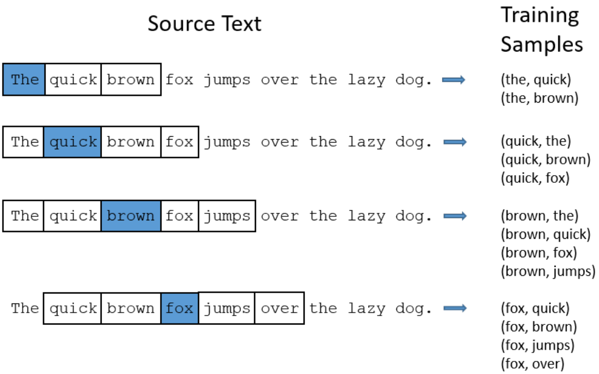



Figure: Figure from Word2Vec Tutorial

# Model: 2. Word Embedding Skip-Gram Model

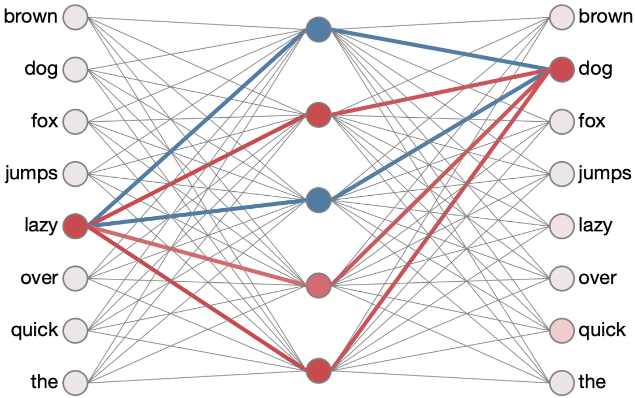



Figure: Figure from Wevi with the specific example. In *gesim*, the default return is the input vector.

# Model: 2. Word Embedding Skip-Gram Model

Let $I$ is the number of words in the corpus, $V$ is the set of all feasible words in the vocabulary, and $v_i$ are $d$-dimensional real-vector word embeddings. $c$ be the window size for the estimation. We select the $v_i$ to maximize:

$$\frac{1}{I}\sum_{i=1}^{I}\sum_{-c\leq j\leq c, j\neq 0}\log p(\text{word}_{i+j} \mid \text{word}_i)$$

where:

$$p(\text{word}_j \mid \text{word}_i) = \frac{\exp(v_j \cdot v_i')}{\sum_{k=1}^{|V|}\exp(v_k \cdot v_i')}.$$

Now we have the vector for each **token**. We concatenate the tokens at the sentence level to obtain the vector representation for a **sentence**:

$$v = [v_1, \ldots, v_n] \in \mathbb{R}^{d\times n}$$

# Model: 2. Word Embedding Skip-Gram Model

After the cleaning and tokenizing, we have[2]:

- ['well', 'really', 'know', 'well', 'pack', 'teledyne', 'piks', 'brj', 'works', 'even_though', 'clearly', 'ordered', 'goodman', 'without', 'notice', 'sent', 'official', 'waterpik', 'pack', 'jt', 'e', 'place', 'total', 'cost', 'paid', 'teledyne', 'pack', 'two', 'piks', 'place', 'four']

Finally, the Word2Vec turns it into a 20 by 1 vector (rounded by 4 digits, averaged over all the tokens for simplicity):

- [0.3674, -0.1955, 0.3926, -0.0273, -0.2642, 0.2046, 0.0489, 0.7512, -0.7463, 0.5247, 0.0127, -0.0863, 0.3502, -0.2463, 0.3582, 0.2546, 0.6332, -0.1017, -0.3856, -0.6695]

[2] The outputs are given by demo_Preprocessing part in ipynb file

# Model 3: CNN Architecture Cont.

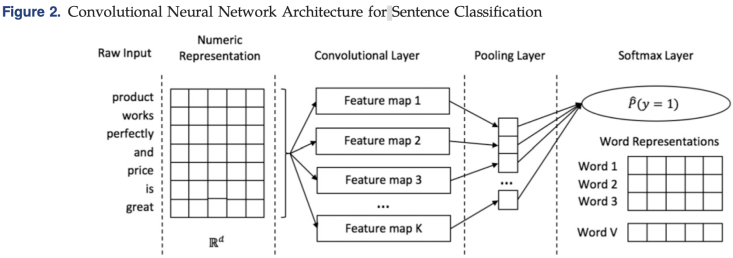



Figure: Figure 2 from Timoshenko and Hauser (2019)

- The word embedding is not hard-coded, but with a *nn.embedding*()
- Padding was done at the end. I use default torch setting to pad on both sides.

For more tech details in the code see the replication files.

# Model 3: Identify Informative Content with CNN

Numerical Representations of Words:

- Words are represented as real-valued vectors $v_i$ from pre-trained embeddings.
- Sentence embedding:

$$v = [v_1, \ldots, v_n] \in \mathbb{R}^{d \times n},$$

Convolution Layer:

- Applies filters $w_t \in \mathbb{R}^{d \times h_t}$ of size $h_t$ to generate feature maps $c^t$.
- Feature computation:

$$c_i^t = \sigma\left(w_t \cdot v_{i:i+h_t-1} + b_t\right),$$

where $v_{i:i+h_t-1} = [v_i, \ldots, v_{i+h_t-1}]$ and $\sigma(x) = \max(0, x)$ is ReLU.

# Model 3: CNN Architecture

Pooling Layer:

- Performs global max pooling to summarize feature maps:

$$z_t = \max(c_1^t, \ldots, c_{n-h_t+1}^t),$$

resulting in $z = [z_1, \ldots, z_r]$, where $r$ is the total number of filters across all filter sizes.

Softmax Layer:

- Outputs probabilities for sentence classification as informative or not:

$$p(y|z) = \text{Softmax}(z \cdot w + b).$$

# Before Replications:

A **direct replication** is hard in this scenario:

- The original code sent by Prof. Timoshenko to me in email was written in Python 2
- Based on an old version of keras for CNN
- np.random.seed(100) is defined in a loop.

So I re-write the codes in the replication file

- Switched the framework to the latest PyTorch under Python 3.10
- Replaced the loop with parallel computing for figure 5 and 7.
- Replaced old stats packages (e.g. fastcluster) with scipy
- For simplicity, I only replicate the **ML-related** sections in the main paper. not the non-ML tables and Appendix.

# Model 4: Sample Diverse Content (with Replication)

Idea:

- We want to reduce the redundant sentences BEFORE e.g. manual consumer need analysis
- The following 3 sentences are considered redundant.

  *"When I am done, my teeth do feel'squeaky clean.'"*

  *"Every time I use the product, my teeth and gums feel professionally cleaned."*

  *"I am still shocked at how clean my teeth feel."*

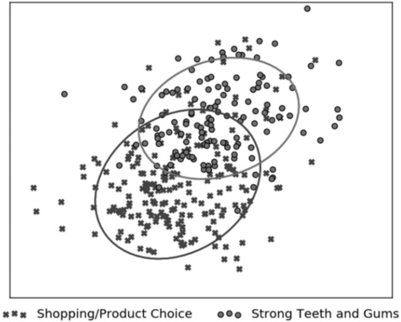
Use sentence embeddings to reduce redundancy

- sampling content for manual review from maximally different parts of the space of sentence embeddings.

# Model 4: Sample Diverse Content (with Replication)

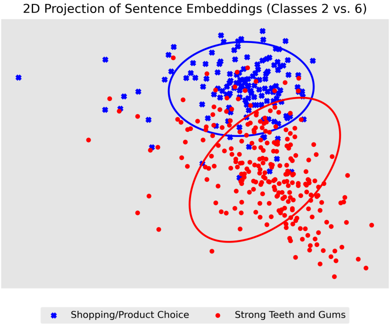
Use the embedding from the Skip-Gram model, apply PCA to 2D by plotting the first 2 principle components.

**Figure 6.** Projections of 20-Dimensional Embeddings of Sentences onto Two Dimensions (PCA)




*Note.* Dots and crosses indicate analyst-coded primary customer needs.

(a) Figure 6




(b) A Replication by Shuyi

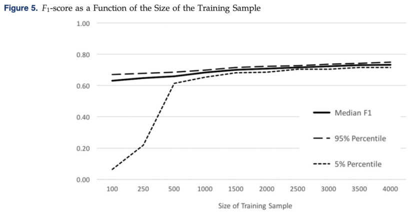
# Model 5 and so: Results and Replications

How good is the CNN model:

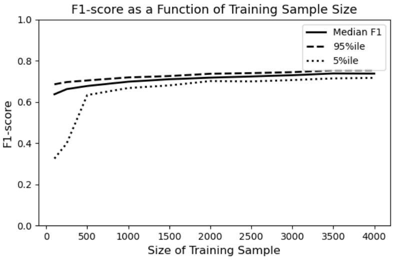
- Given different training sample?
- Compared with different models?

# Model 5 Fig 5: Results and Replications

How good is the CNN model?




(a) Figure 5




(b) A Replication by Shuyi

*"Performance of the CNN stabilizes after 500 training sentences, with some slight improvement after 500 training sentences."*

# Model 5 Table 2: Results and Replications Cont

How good is the CNN model?

**Table 2.** Alternative Machine-Learning Methods to Identify Informative Sentences

| Method | Precision (%) | Recall (%) | Accuracy (%) | $F_1$ (%) |
|---|---|---|---|---|
| Convolutional neural network (CNN) | 74.4 | 73.6 | 74.2 | 74.0 |
| CNN with asymmetric costs ($\gamma = 3$) | 65.2 | 85.3 | 70.0 | 74.0 |
| Recurrent neural network-LSTM | 72.8 | 74.0 | 73.2 | 73.4 |
| Multichannel CNN | 70.5 | 74.9 | 71.8 | 72.6 |
| Support vector machine | 63.7 | 67.9 | 64.6 | 65.7 |

(a) Table 2

```
Results:
Model              Precision    Recall    F1        Accuracy
CNN (gamma = 1)    0.7500       0.7104    0.7297    0.7332
CNN (gamma = 3)    0.6548       0.8933    0.7557    0.7073
LSTM Network       0.7682       0.7301    0.7487    0.7516
SVM                0.7151       0.7296    0.7223    0.7157
(ds_torch) (base) liushijian@10-21-196-38 code %
```

(b) A Replication by Shuyi

- Results align with the original paper.
- Fail to run the Kim et al. 2014 Multichannel CNN as the tensor shape is tricky to align.

# Model 5 Figure 7: Results and Replications Cont

Compare content selection approaches in terms of the **expected number of unique customer needs identified** in Y sentences.

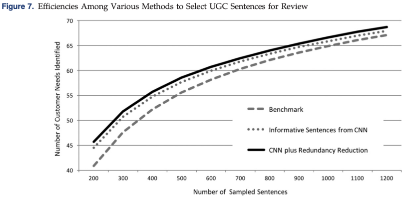
- Unique Customer Needs: identified_needs.csv
- (1) randomly split the 6,700 preprocessed sentences, which are neither too short nor too long, into 3,700 training and 3,000 holdout samples
- 2) train the CNN using the training sample
- 3) draw Y sentences from the holdout sample for review. We count the unique needs identified in the Y sentences and repeat the process 10,000 times.

Three methods:

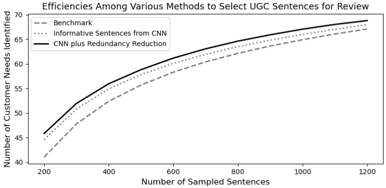
- Benchmark: Do nothing. Sample and count.
- Info Sentences CNN: Use CNN to identify informative sentences; sample from informative sentences for review.
- Redundancy Reduction: Sentence-embedding clusters to reduce redundancy after Info CNN [3]

[3] In the code the authors then sample from each cluster (in total max_clusters generated by Unsupervised learning)

# Model 5 Figure 7: Results and Replications Cont




(a) Figure 7




(b) A Replication by Shuyi

*"The CNN improves efficiency as indicated by the dotted line. Using the CNN and clustering sentence embeddings increases efficiency further."*

# Contributions

Empirical:

- Challenges the traditional Voice-of-Customer (VOC) methods. UGC can be an equally or more effective source for identifying customer needs.
- Empirical evidence: UGC + ML + Analyst captures a broader and more diverse set of customer needs compared to traditional methods.
- Very **low cost**, very effective, with very **simple structure**.

Methodology:

- Introduces a CNN-based classifier to filter out non-informative content from UGC.
- Utilizes sentence embeddings to cluster and reduce redundancy.
- "Human-AI Interaction"?

# Limitations

Methodology:

- Explicitly perform word embedding and the CNN architecture are somehow "out of date" Vaswani (2017)
  - Restrictions on input length (Very long customer reviews can be useful)
  - Position encoding & End-to-end embedding.
  - Unnecessary to remove stop words and numbers beforehand.
- May not work for consumer needs search for innovative and new products. (VOC?)
  - Only works for firms with large UGC e.g. huge oral care product reviews
  - The economics of scale v.s. VOC.
- Do we really need to train a model for ourselves? Or can we solve it with LLM?

Extensions:

- Other sources of UGC: e.g. Stream Videos?
- Combined analysis: What are the "most urgent" consumer needs that e.g. result in frequent returns?

# Limitations

## Why not just LLM?

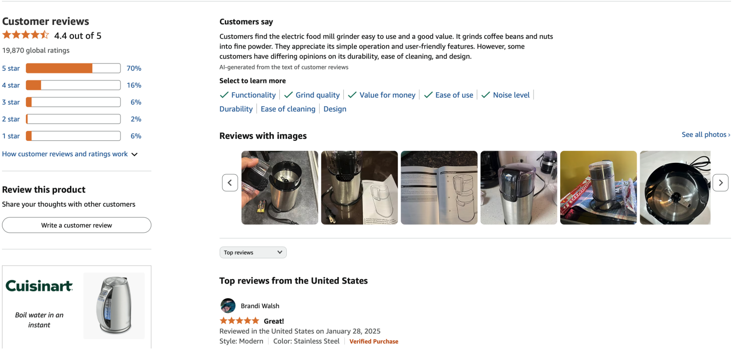



Figure: LLM Summary Example from Amazon

# New Methods: Why not just ChatGPT

> *Imagine you are a marketing expert. Now I am giving you two datasets with user generated contents on the oral products.. The "Sentence Text" are the user generated reviews, the "isNeed" is a dummy variable that equals 1 if the "Sentence Text" is talking about a specific need that the consumer wants. Now, please read the "train_4000.csv" carefully. Please fully understand what does it mean for an review to represent consumer need. Do you understand the task?*
>
> *Great. Now I will give you another file. Please read this file carefully, and add a new column called isNeed_res that equals 1 if a sentence is about a specific need, and 0 otherwise. Please use the knowledge you obtained from the previous file. Please return a csv file with only two columns: "Sentence ID" and "isNeed_res".*

Train/Test = 1
Accuracy: 0.48325
Probably not that easy.

# References I


Timoshenko, A., & Hauser, J. R. (2019, January). Identifying Customer Needs from User-Generated Content. *Marketing Science*, *38*(1), 1–20. Retrieved 2025-01-31, from `https://pubsonline.informs.org/doi/10.1287/mksc.2018.1123` doi: 10.1287/mksc.2018.1123

Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.