Lifelong Domain Word Embedding via Meta-Learning

Hu Xu, University of Illinois at Chicago
Bing Liu, University of Illinois at Chicago
Lei Shu, University of Illinois at Chicago
Philip S. Yu, University of Illinois at Chicago, Tsinghua University

IJCAI ’18

Motivation

- Learning high-quality domain word embeddings is important for achieving good performance in many NLP tasks.
- Two dilemmas:
  - General-purpose embeddings trained on large-scale corpora are often sub-optimal for domain-specific applications.
  - Domain-specific tasks often do not have large in-domain corpora for training high-quality domain embeddings.

Ideas

- Can we filter a large-scale embedding corpus so when building domain embeddings we only use a partial “relevant” corpus ?
  - For example, if we want ``java” vector in a programming domain, we only use corpus from in the context such as ``function'' or ``variable'' but not coffee shop.

- Can we make this process continuous ? So new domain corpus can be easily added and better embeddings can be built if we have many domain corpora.

Roadmaps

- **Problem Statements, Lifelong Learning**
- Meta-Learning as a Solution
- L-DEM and Meta-Learner
- Experiments
- Conclusions

Problem Statements

Problem statement: We assume that the learning system has seen n domain corpora in the past: $D_{1:n} = \{D_1, \ldots, D_n\}$, when a new domain corpus D_{n+1} comes with a certain task, the system automatically generates word embeddings for the $(n+1)$-th domain by leveraging some useful information or knowledge from the past n domains.

Lifelong Machine Learning

- This problem definition is in the **lifelong learning** (LL) setting, where the new or (n+1)-th task is performed with the help of the knowledge accumulated over the past n tasks (Chen and Liu, 2016).

Roadmaps

- Problem Statements, Lifelong Learning
- **Meta-Learning as a Solution**
- L-DEM and Meta-Learner
- Experiments
- Conclusions

Challenge

- How to automatically identify relevant information from the past n domains with no user help
-

Meta-Learning as a Solution

- The goal of meta-learning is to understand how automatic learning can become flexible in solving learning problems.

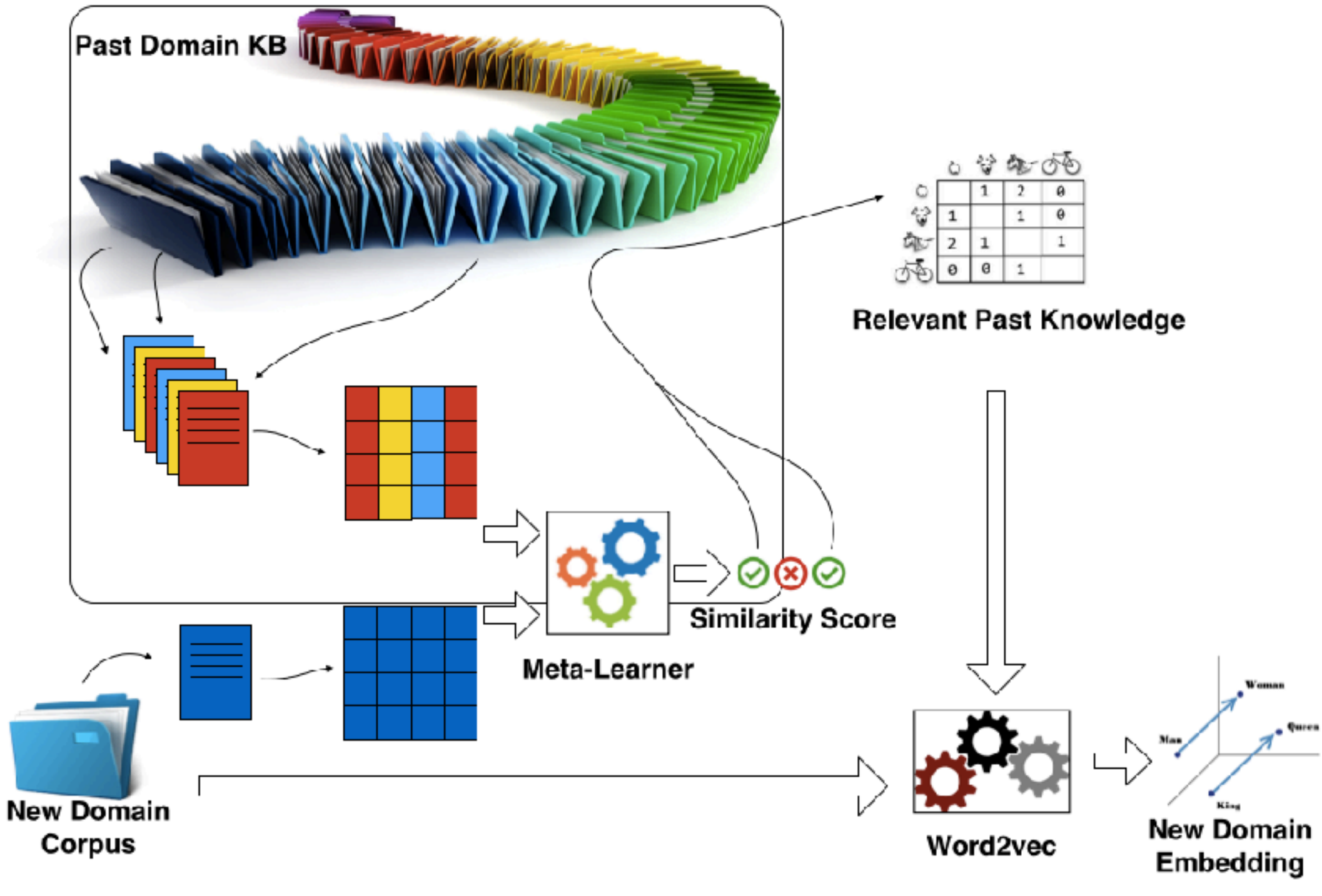
- We train a meta-learner (model) so to smartly identify relevant corpus from past domains to help learn better domain embeddings.

- We propose a meta-learning based system L-DEM (Lifelong Domain Embedding via Meta-learning) to tackle the challenge.

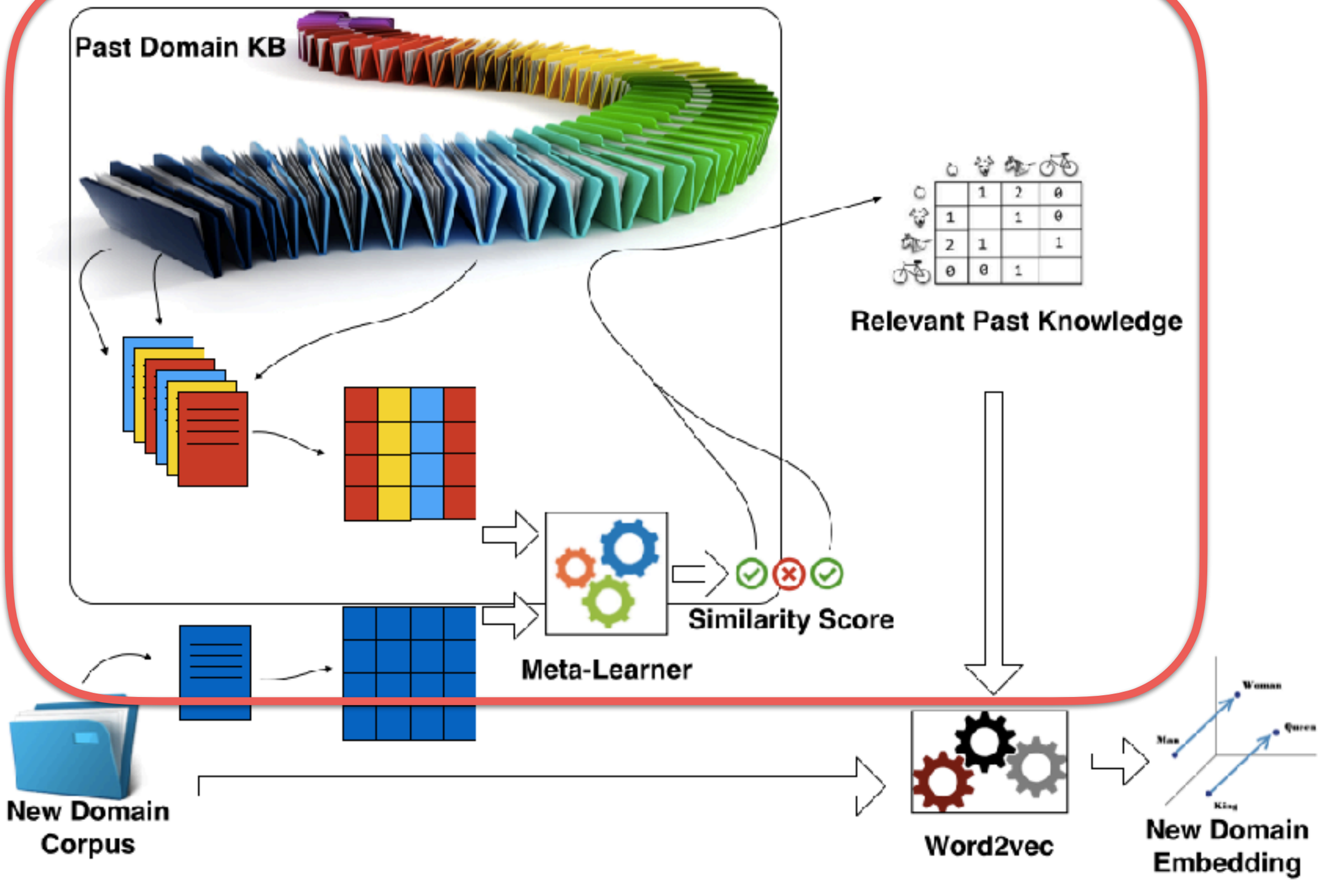
Roadmaps

- Problem Statements, Lifelong Learning
- Meta-Learning as a Solution
- **L-DEM and Meta-Learner**
- Experiments
- Conclusions

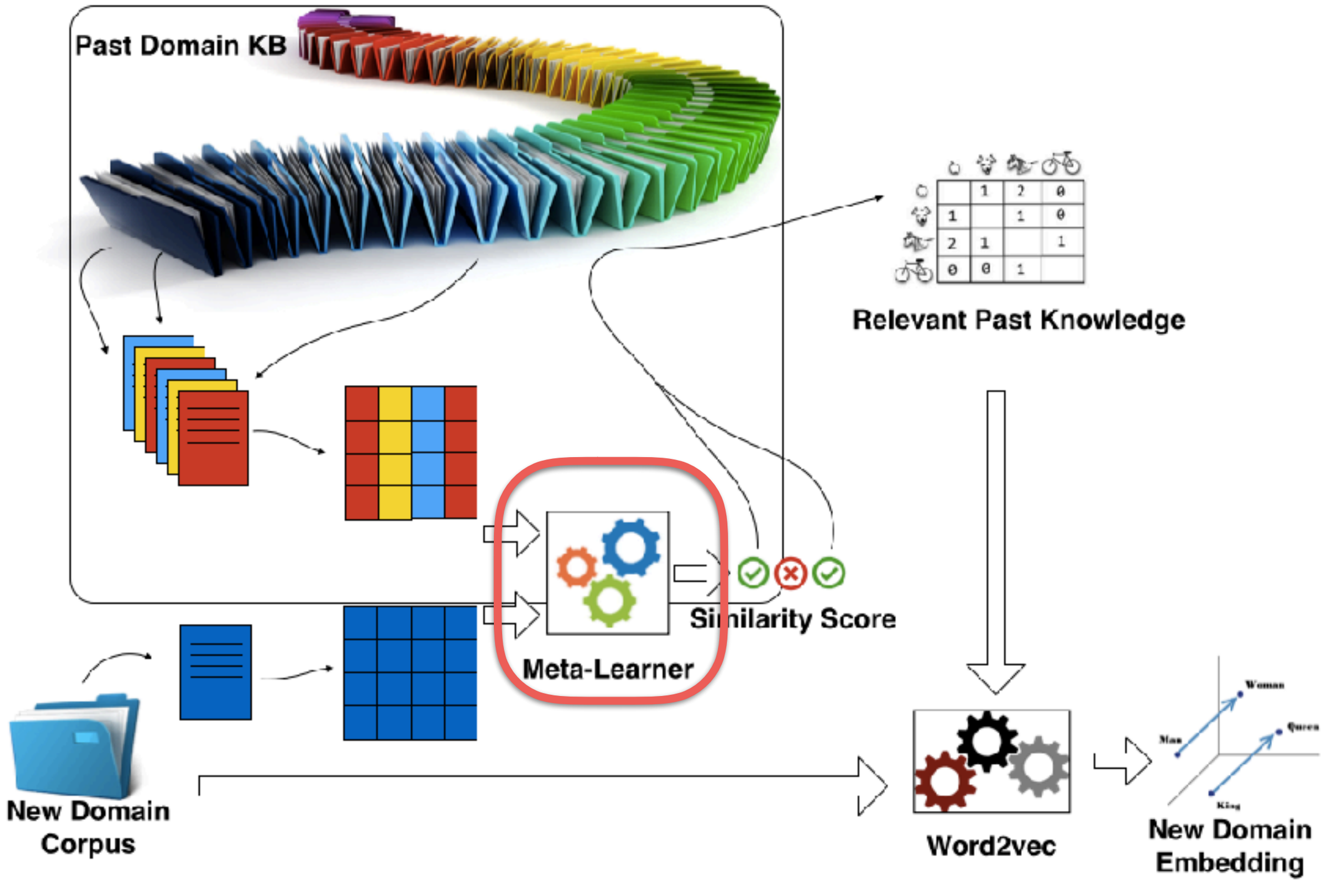
L-DEM


Past Domain KB
Relevant Past Knowledge
Similarity Score
Meta-Learner
New Domain
Corpus
Word2vec
Woman
Man
Queen
King
New Domain
Embedding

L-DEM


Past Domain KB
Relevant Past Knowledge
Similarity Score
Meta-Learner
New Domain
Corpus
Word2vec
Woman
Man
Queen
King
New Domain
Embedding

L-DEM


Past Domain KB
Relevant Past Knowledge
Meta-Learner
Similarity Score
New Domain
Corpus
Word2vec
Woman
Man
Queen
King
New Domain
Embedding

Retrieve Relevent Past Knowledge

Algorithm 1: Identifying Context Words from the Past

Input : a knowledge base $\mathcal{K}$ containing a vocabulary $\mathcal{K}.V_{wf}$, a base meta-learner $\mathcal{K}.M$, and domain knowledge $\mathcal{K}_{m+1:n}$;
a new domain corpus D_{n+1}.

Output: relevant past knowledge $\mathcal{A}$, where each element is a key-value pair $(w_t, \mathcal{C}_{w_t})$ and $\mathcal{C}_{w_t}$ is a list of context words from all similar domain contexts for w_t.

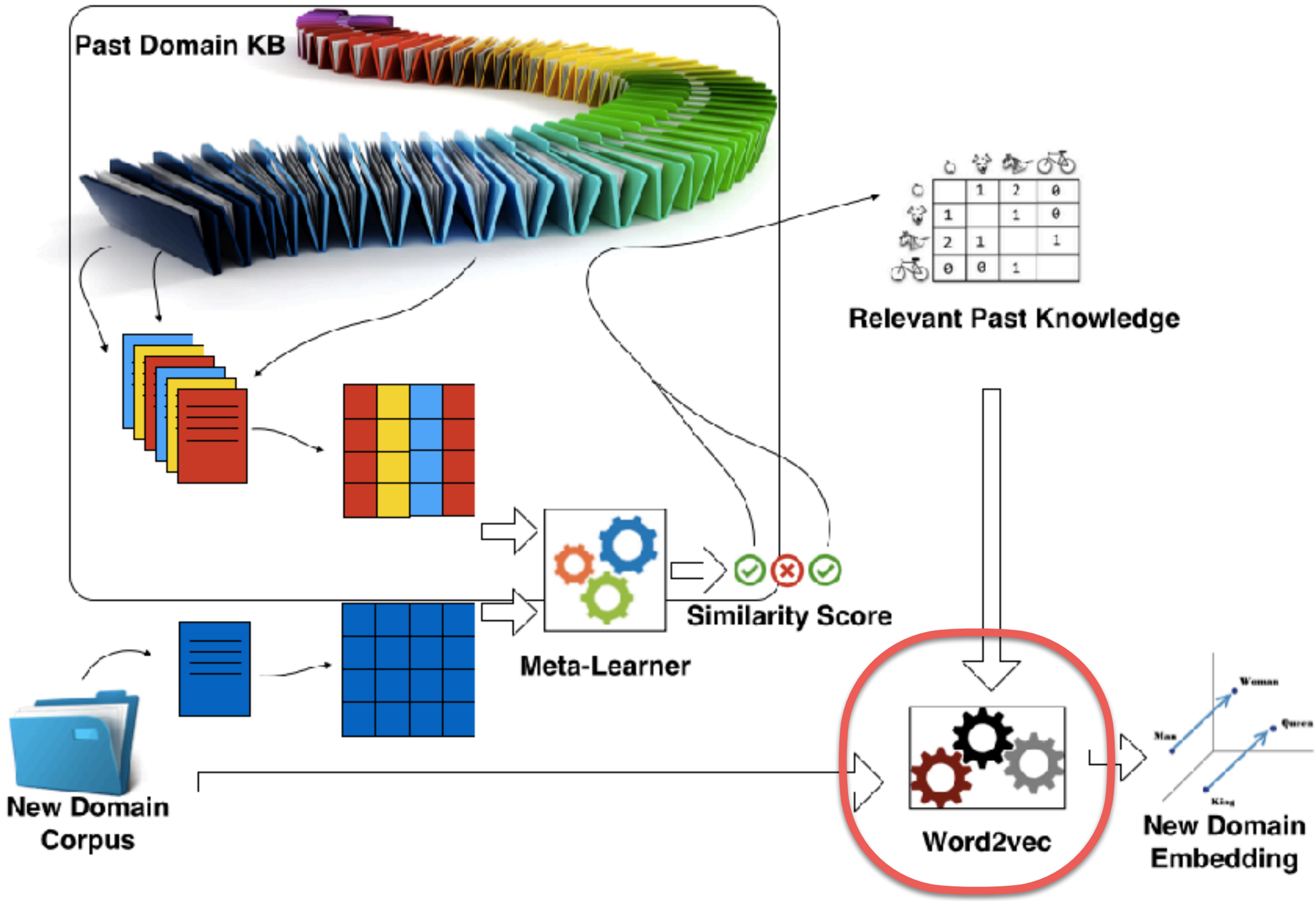
$(V_{m+1:n}, C_{m+1:n}, E_{m+1:n}) \leftarrow \mathcal{K}_{m+1:n}$
$V_{n+1} \leftarrow \text{BuildVocab}(D_{n+1})$
$C_{n+1} \leftarrow \text{ScanContextWord}(D_{n+1}, V_{n+1})$
$E_{n+1} \leftarrow \text{BuildFeatureVector}(D_{n+1}, \mathcal{K}.V_{wf})$
$M_{n+1} \leftarrow \text{AdaptMeta-learner}(\mathcal{K}.M, E_{m+1:n}, E_{n+1})$
$\mathcal{A} \leftarrow \emptyset$
for $(V_j, C_j, E_j) \in (V_{m+1:n}, C_{m+1:n}, E_{m+1:n})$ **do**
  $O \leftarrow V_j \cap V_{n+1}$
  $F \leftarrow \{(\mathbf{x}_{o,j,1}, \mathbf{x}_{o,n+1,1}) |$
    $o \in O$ and $\mathbf{x}_{o,j,1} \in E_j$ and $\mathbf{x}_{o,n+1,1} \in E_{n+1}\}$
  $S \leftarrow M_{n+1}.\text{inference}(F)$
  $O \leftarrow \{o | o \in O \text{ and } S[o] \geq \delta\}$
  for $o \in O$ **do**
    $\mathcal{A}[o].\text{append}(C_j[o])$
  end
end
$\mathcal{K}_{n+1} \leftarrow (V_{n+1}, C_{n+1}, E_{n+1})$
return $\mathcal{A}$

Meta-Learner

- Train a siamese network.
- Given a pair of **x**s representing the same word from two domains, detect whether two **x**s have relevent meanings.
- This network is trained from a holdout m domains

$$\sigma\left(\boldsymbol{W}_2 \cdot \mathbf{abs}\left(\left(\boldsymbol{W}_1 \cdot \frac{\mathbf{x}_{w_{i,j,k}}}{|\mathbf{x}_{w_{i,j,k}}|_1}\right) - \left(\boldsymbol{W}_1 \cdot \frac{\mathbf{x}_{w_{i,j',k'}}}{|\mathbf{x}_{w_{i,j',k'}}|_1}\right)\right) + b_2\right), \quad (1)$$

L-DEM


Past Domain KB
Relevant Past Knowledge
Meta-Learner
Similarity Score
New Domain
Corpus
Word2vec
Woman
Man
Queen
King
New Domain
Embedding

Augmented Embedding Training

- A modified version of word2vec so to take two inputs: the new domain corpus and relevent past knowledge ***A***.

$$\mathcal{L}_{D_{n+1}} = \sum_{t=1}^{T} \Big(\sum_{w_c \in \mathcal{W}_{w_t}} \big(\log \sigma(\boldsymbol{u}_{w_t}^T \cdot \boldsymbol{v}_{w_c}) + \sum_{w_{c'} \in \mathcal{N}_{w_t}} \log \sigma(-\boldsymbol{u}_{w_t}^T \cdot \boldsymbol{v}_{w_{c'}}) \big) \Big), \tag{2}$$

$$\mathcal{L}_{\mathcal{A}} = \sum_{(w_t, \mathcal{C}_{w_t}) \in \mathcal{A}} \Big(\sum_{w_c \in \mathcal{C}_{w_t}} \big(\log \sigma(\boldsymbol{u}_{w_t}^T \cdot \boldsymbol{v}_{w_c}) + \sum_{w_{c'} \in \mathcal{N}_{w_t}} \log \sigma(-\boldsymbol{u}_{w_t}^T \cdot \boldsymbol{v}_{w_{c'}}) \big) \Big). \tag{3}$$

$$\mathcal{L}'_{D_{n+1}} = \mathcal{L}_{D_{n+1}} + \mathcal{L}_{\mathcal{A}}. \tag{4}$$

-

Roadmaps

- Problem Statements, Lifelong Learning
- Meta-Learning as a Solution
- L-DEM and Meta-Learner
- **Experiments**
- Conclusions

Experiment Settings

- We use the performances of a classification task to evaluate the proposed method.

- We use the Amazon Review datasets (He and McAuley, 2016), which is a collection of multiple-domain (we take a 2nd-level category of a product as a domain) corpora.

- We first randomly select m=56 domains to train and evaluate the base meta-learner.

- Then from rest domains, we sample three random collections with 50, 100 and 200 (n-m) domains corpora, respectively, as three settings of past domains.

Experiment Settings

- We further randomly selected 3 new testing domains: Computer Components (CC), Kitchen Storage and Organization (KSO) and Cats Supply (CS) and formulate 3 review classification tasks.

- These give us three text classification problems, which have 13, 17, and 11 classes respectively.

- We set the size of the new domain corpora to be 10 MB and 30 MB to test the performance in the two sizes of new domains.

Evaluation of Meta-Learner

- We split the first 56 domains into 39 domains for training, 5 domains for validation and 12 domains for testing. So the validation and testing domain corpora have no overlap with the training domain corpora.

- We sample 2 sub-corpora for each domain so we are able to form positive examples.

- We randomly select 2000, 500, 1000 words from each training domain, validation domain, and testing domain, respectively.

- This ends up with 80484 training examples, 6234 validation examples, and 20740 test examples.

- F1-score of the proposed base meta-learner model is **0.81**.

Evaluation of Meta-Learner

- We split the first 56 domains into 39 domains for training, 5 domains for validation and 12 domains for testing. So the validation and testing domain corpora have no overlap with the training domain corpora.

- We sample 2 sub-corpora for each domain so we are able to form positive examples.

- We randomly select 2000, 500, 1000 words from each training domain, validation domain, and testing domain, respectively.

- This ends up with 80484 training examples, 6234 validation examples, and 20740 test examples.

- F1-score of the proposed base meta-learner model is **0.81**.

Evaluation of Adapted Meta-Learner

- We sample 3000 words from each new domain and select 3500 paired examples for training, 500 examples for validation and 2000 examples for testing.

	CC	KSO	CS
10MB	0.832	0.841	0.856
30MB	0.847	0.859	0.876

Table 1: F1-score of positive predictions of the adapted meta-learner on 3 new domains: Computer Components (CC), Kitchen Storage and Organization (KSO) and Cats Supply (CS).

-

Classification Tasks

- We have classification tasks on 3 new domains.
- We randomly draw 1500 reviews from each class to make up the experiment data, from which we keep 10000 reviews for testing and split the rest 7:1 for training and validation, respectively.
- We train and evaluate each task on each system 10 times (with different initializations) and average the results.

Classification Model

- We use a simple Bi-LSTM model followed by a softmax layer to evaluate the performance of different embeddings.

- The input size of Bi-LSTM is the same as the embedding and the output size is 128.

- We apply dropout rate of 0.5 on all layers except the last layer and use Adam as the optimizer.

Baselines

- No Embedding (NE)
- fastText (Wiki.en), GoogleNews
- GloVe.Twitter.27B, GloVe.6B, GloVe.840B
- New Domain 10M (ND 10M), New Domain 30M (ND 30M)
- 200 Domains + New Domain 30M (200D + ND 30M)
- L-DENP 200D + ND 30M}: This is a Non-Parametric variant of the proposed method. We use TFIDF as the representation for a sentence in past domains and use cosine as a non-parametic function to compute the similarity with the TFIDF vector built from the new domain corpus.
- L-DEM Past Domains + New Domain (L-DEM [P]D + ND [X]M)

Results

	CC(13)	KSO(17)	CS(11)
NE	0.596	0.653	0.696
fastText	0.705	0.717	0.809
GoogleNews	0.76	0.722	0.814
GloVe.Twitter.27B	0.696	0.707	0.80
GloVe.6B	0.701	0.725	0.823
GloVe.840B	0.803	0.758	0.855
ND 10M	0.77	0.749	0.85
ND 30M	0.794	0.766	0.87
200D + ND 30M	0.795	0.765	0.859
L-DENP 200D + ND 30M	0.806	0.762	0.870
L-DEM 200D + ND 10M	0.791	0.761	0.872
L-DEM 50D + ND 30M	0.795	0.768	0.868
L-DEM 100D + ND 30M	0.803	0.773	0.874
L-DEM 200D + ND 30M	**0.809**	**0.775**	**0.883**

Table 2: Accuracy of different embeddings on classification tasks for 3 new domains (numbers in parenthesis: the number of classes)

Conclusions

- We formulate a domain word embedding learning process.

- Given many previous domains and a new domain corpus, the proposed method can generate new domain embeddings by leveraging the knowledge in the past domain corpora via a meta-learner.

- Experimental results show that our method is highly promising.