Knowledge Graph Embeddings

Minglan Zheng

December 7, 2021

Contents

1 Overview of the report **2**

2 Introduction **2**

2.1 Why do we need Knowledge Graph Embedding 2

2.2 Anatomy of a Knowledge Graph Embedding Model 2

2.3 Related Work 3

2.3.1 ComplEx 3

2.3.2 RotatE 4

2.4 Mathematical Theory of Quaternion KGE 4

3 Proposed Quaternion Kernel **6**

4 Experiments and Discussion **6**

4.1 Link Prediction & Triangle Classification 7

4.2 Semantic Analysis 8

6 Citation **10**

1 Overview of the report

Section [2.1 - 2.2] gives an overview of the background and problem formulation, section [2.3] reviews related techniques in literature, section [2.4] motivate the theory behind Quaternion knowledge graph embedding and explain the method in details, Section [3] examines the mathematics of proposed Quaternion kernel scoring function. Section [4] analyzes experiment results. This report focuses more on the theoretical reasoning behind each designed approach and analysis of experimental results. For implementation and information about replicating results, please refer to the github page.

2 Introduction

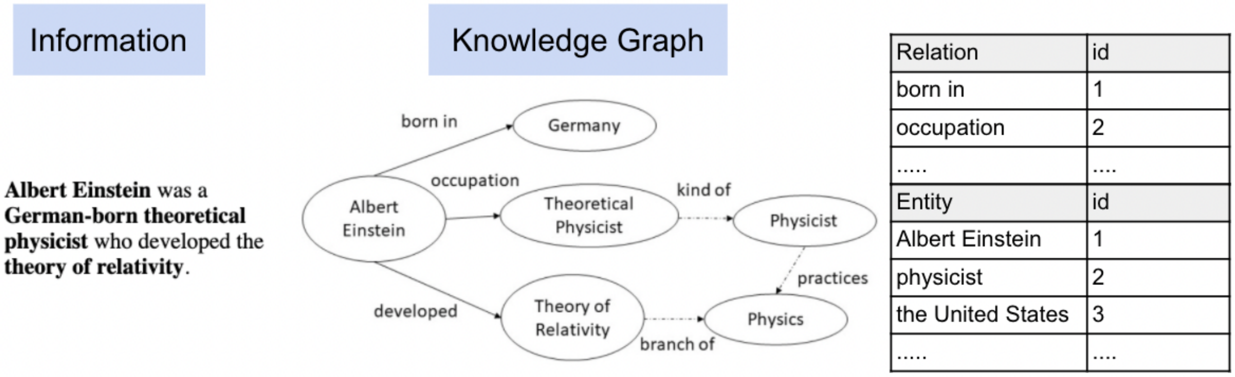
The study of graph representation learning converts nodes and edges into vector embeddings, to better leverage the wealth of neural architectures designed to process vectors. The method described in the report is Knowledge Graph Embeddings (KGE), a supervised learning task that maps entities and relations into a continuous low-dimensional space.

2.1 Why do we need Knowledge Graph Embedding

Before introducing the techniques, this section gives an overview of why we study this concept. *Knowledge graph (KG) embedding enables real-world applications to consume information*. KG is a multi-relational graph in forms of entities and relations. In literature, the directed edges of knowledge graph is often represented as triplets, i.e., $< h, r, t >$, whereas $h, t \in E$ (entities) and $r \in R$ (relations). It is difficult for the real-world applications, for example those that rely on deep machine learning algorithms, to directly consume the information in its symbolic triplets format, whereas it is more versatile to consume and manipulate if the graph is converted to a numerical, i.e, vector, representation.

2.2 Anatomy of a Knowledge Graph Embedding Model

Knowledge graph embedding (KGE) is an active research area. At the core of KGE design is to optimize a scoring function that assigns a score proportional to the likelihood that an unseen triple (s, p, o) is true. A good scoring function captures properties of the graph including symmetric/asymmetric, inversion, composition. *In Section 2.3, We will review two types of scoring function in literature to motivate the proposed Quaternion Knowledge Graph Embedding.*



Relation	id
born in	1
occupation	2
.....	....
Entity	id
Albert Einstein	1
physicist	2
the United States	3
.....	....

Figure 1: An example of a knowledge graph extracted from the sentence on the left to illustrate formulating the training of KG embedding as a classification problem with labels -1 and 1, i.e., <Albert Einstein, born in, Germany> has a label of 1.
Image credit: Stanford University CS 520.

Training such embedding involves optimizing the parameters (i.e., embeddings vector). In the representation, triples can be represented by a binary value $Y \in \{-1, 1\}$ to denote a set of true and false facts. There are several training tricks, one is to train with synthetic negative triples. For example, $C = \{(\hat{s}, p, o) \mid \hat{s} \in \varepsilon\} \cup \{(s, p, \hat{o}) \mid \hat{o} \in \varepsilon\}$ corrupts the subject $\hat{s}$ or the object $\hat{o}$ of the positive triples. A similar trick is to inject reciprocal triples into the training set to reinforce properties of symmetry and asymmetry. To illustrate, suppose the following graph in Figure 1 is scraped from Wikipedia, and the charts define entity and relation to id mapping. Then <1, 1, 3> has a label of 1 to denote it exists in the graph. A corrupted triple <1, 1, 3> or a reciprocal triple <3, 1, 1> with the label of -1 since "born in" is an anti-symmetric relation.

One thing to note, the position of the vector representation of nodes is semantically meaningful. To illustrate, nodes of similar concepts tend to have their vector representations in the same portion of the embedding space. We leverage this property to perform an experiment on the downstream task of semantic analysis in Section 4.2.

2.3 Related Work

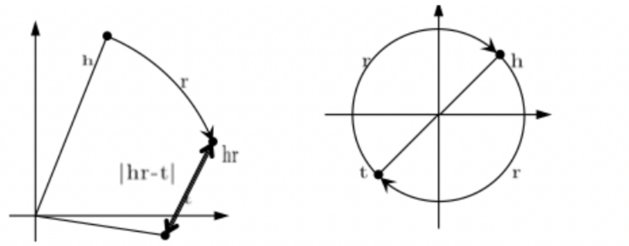
Recent trends in knowledge graph embeddings proposed to use models that operate in the complex space, instead of euclidean space (real n-space R^n) , to overcome the challenges in representing relational properties. In this section, we will provide a summary of two of these models, ComplEx and RotatE.

2.3.1 ComplEx

ComplEx utilizes Hermitian dot-product to model the antisymmetric patterns. With complex number, the dot product, Hermitian product, is defined as $< u, v > := \bar{u}^T v$.

Since this dot product involves the conjugate-transpose of one of the two vectors, it is **not** symmetric, and can effectively capture antisymmetric relations. In ComplEx, u, v are complex vectors, such that $u = Re(u) + iIm(u)$.

2.3.2 RotatE



(b) RotatE models **r** as rotation in complex plane.

(c) RotatE: an example of modeling symmetric relations **r** with $r_i = -1$

Figure 2: Illustration of RotatE with 1 dimension of embedding. Image Credit (Sun et al.)

RotatE is the first model capable of predicting symmetric/anti-symmetric, inversion and composition relation. Compared to the earlier transitional model TransE, RotatE is superior in its capability to model symmetric relations. Different from ComplEx, RotatE introduces composition relations. Motivated by Euler's identity, $e^{i\theta} = cos(\theta) + isin(\theta)$, RotatE represent the relation between h and t as angle of rotation: $t = h \circ r$ (shown by Figure 2(b). By restricting $|r_i| = 1$, we are in the unit circle, as shown by Figure 2(c), and the relation is symmetric $r(x, y) \Rightarrow r(y, x)$. Composition relation $r_2(x, y) \wedge r_3(y, z) \Rightarrow r_1(x, z)$ can defined as $r_1 = r_2 \circ r_3$ such as for $\theta_1 = \theta_2 + \theta_3$, r_1 is obtained by a combined rotation of r_2 and r_3.

2.4 Mathematical Theory of Quaternion KGE

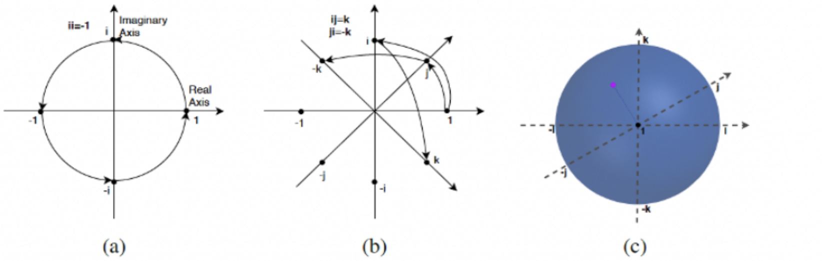


Figure 3: (a) Complex plane; (b) Quaternion units product; (c) sterographically projected hypersphere in 3D space.
Image credit (original Figure 1 from Quaternion Knowledge Graph Embedding by Zhan et al.)

In this section, we explain the mathematical backgrounds related to advantages of Quaternion embedding.
Quaternion is in the hypercomplex space, consisting of a real and three imaginary components (2 more than usual complex numbers), defined as $Q = a + bi + cj + dk$, $s.t. i^2 = j^2 = k^2 = ijk = -1$. The QuatE framework subsumes the ComplEx method. This can be easily shown by setting two of the imaginary units to zero. Therefore, it inherits all the benefits of ComplEx, including its ability to model symmetry, anti-symmetry, and inversion. The intuition behind more degrees of freedom is to enable expressive rotation and better geometrical interpretations. We will dive deeper

into other advantages compared to operations in Euler space such as *avoiding gimbal lock* after reviewing some Quaternion algebra and the proposed algorithm.

Inner Product involves summing inner products between corresponding scalar and imaginary components.

$$Q_1 \cdot Q_2 \ = \ < a_1, a_2 > \ + \ < b_1, b_2 > \ + \ < c_1, c_2 > \ + \ < d_1, d_2 > \qquad (2)$$

Norm: $|Q| \ = \ \sqrt{a^2 + b^2 + c^2 + d^2}$ (3)

Hamilton Product (Quaternion Multiplication) (4)

$$Q_1 \otimes Q_2 = (a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2) + (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)\mathbf{i} + (a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)\mathbf{j} + (a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)\mathbf{k}’$$

Head, tail entities and relation are represented by quaternion as shown below.

Head entity $Q_h = \{a_h + b_h\mathbf{i} + c_h\mathbf{j} + d_h\mathbf{k} : a_h, b_h, c_h, d_h \in \mathbb{R}^k\}$

Tail entity $Q_t = \{a_t + b_t\mathbf{i} + c_t\mathbf{j} + d_t\mathbf{k} \ : a_t, b_t, c_t, d_t \in \mathbb{R}^k\}$

Relation $W_r = \{a_r + b_r\mathbf{i} + c_r\mathbf{j} + d_r\mathbf{k} \ : a_r, b_r, c_r, d_r \in \mathbb{R}^k\}.$

Before the main methods, a step of relation normalization is necessary, $W_r^{\triangleleft} = \frac{W_r}{|W_r|}$ (5)

The author explains this because the scaling effects in non unit quaternions are detrimental. Empirically, removing this normalization step results in significant performance reduction (MRR) of 16% - 23% on standard datasets such as WN18 and WN18RR, respectively. The proposed method is mainly composed of two steps. **<u>First</u>**, rotate the head quaternion using the unit relation quaternion calculated by (5), $Q_h' \ = \ Q_h \ \otimes \ W_r^{\triangleleft}$ (6). By this, latent inter-dependencies are captured with the Hamilton product. *Two properties of Hamilton products are worth noting*. First, they are *asymmetric*, similar to the Hermitian product used in ComplEx, which is a desired property when modeling directed edges. Secondly, it achieves *smooth* three-dimensional rotations and is easier to compose.

<u>Next</u>, the score is a scalar calculated by taking the quaternion inner product between the rotated head quaternion calculated by (6) and the tail quaternion to score each triplet, denoted by equation (7). If a triplet exists in the KG, the angle between the rotated head and tail entity is small, and the inner product will produce a maximized score.

$$\phi(h, r, t) = Q_h' \cdot Q_t = \langle a_h', a_t \rangle + \langle b_h', b_t \rangle + \langle c_h', c_t \rangle + \langle d_h', d_t \rangle \qquad (7)$$

Lastly, similar to ComplEx (Trouillon et al.), this method use a classification loss (regularized logistic loss),

$$L(Q,W) = \sum_{r(h,t)\in\Omega\cup\Omega^{-}} \log(1+\exp(-Y_{hrt}\phi(h,r,t))) + \lambda_1 \parallel Q \parallel_2^2 + \lambda_2 \parallel W \parallel_2^2 \tag{8}$$

The L2 regularization terms add a penalty on the loss function. This is crucial to prevent overfitting (memorizing the data) for complex models in machine learning.

Besides the advantages mentioned above, representing rotations in Quaternion space rather than Euler's space can avoid gimbal lock. For example, let the three rotations known as pitch, roll and yaw. Gimbal lock occurs when two rotation axes align, which results in loss of one degree of freedom and ambiguity arises. The only way to avoid gimbal lock is to use quaternion to introduce a forth gimbal.

3 Proposed Quaternion Kernel

One of the main advantages of learning embeddings in Quaternion space mentioned in the paper is the higher degree of freedom. Following this line of thought, I proposed a modification to the original Quaternion KGE paper, and this section analyzes the effects of utilizing a Quaternion Kernel in the scoring function. In summary, this implementation has two advantages: (1) it enables operation in higher dimensional space and, therefore, adds complexity to the scoring function (2) empirically faster training time, as shown in Figure 4.

The kernel trick is introduced as an extension of the original SVM algorithm for nonlinear classification. Real-valued gaussian and polynomial kernels typically consider quaternion kernel estimation is still an emerging field. To design a Quaternion Kernel scoring function, I consulted the paper Quaternion Reproducing Kernel Hilbert Spaces by Tobar and Mandic and adapted function (2) to function (9) as follows.

$$K_{RG}(Q_h{}', Q_t) = \exp(-A_r\, (Q_h{}' - \ Qt)^H (Q_h{}' - Qt)),\ A_r > 0 \tag{9}$$

Gaussian kernel serves as a deviation measure between samples, hence, K_{RG} is suited for interpolation applications as it can be regarded as a measure of similarity of samples, which indeed seems fit to the original intention of measuring the similarity of the rotated head and tail entity.

4 Experiments and Discussion

Experiment Setup: We conducted experiments on three widely used benchmarks: WN18RR (4.2), and FB15K(4.1). WN18RR (Dettmers et al.,2018) is a subset of WordNet with inverse relations removed. WN18RR contains 11 relations. Its *entities* consist of an exhaustive list of verbs and nouns but lack prepositions (i.e., 'a', 'on') since the English meaning behind prepositions is captured by *relations*. WN18RR's entities also do not contain the names of people. These characteristics are critical for explaining the results of using Quaternion Embedding trained on WN18RR for the semantic classification task on AGNews in section 4.2. The FB15k dataset contains knowledge base relation triples and textual mentions of Freebase entity pairs.

4.1 Link Prediction & Triangle Classification

Link prediction is a way of Knowledge graph augmentation (Paulheim, 2017). Real-world Knowledge graphs are usually incomplete, as they are scraped from large scale websites such as Wikipedia, WordNet, Facebook. Link prediction infers missing information based on the already existing relational data. Triple Classification is the problem of identifying whether a given triple is correct. During training, the hyperparameters (embedding size, regularization rate, and the hyperparameter for kernel definition Ar) are determined by grid search. Because we only train on a single GPU, and the purpose of this experiment is for comparison instead of benchmarking, we stop training for each hyperparameter setting at 50 epochs. The optimal results are reported in Figure 5 and Figure 6.

One thing to note, the advantage that the Quaternion kernel scoring function shows in training time. As illustrated in Figure 4, the Quaternion kernel implementation is about 2 times faster than its Quaternion dot-product counterpart throughout the training process. We know that using Kernel finds the dot product (similarity) in infinite dimension without an increase in computation. However, the reduction in computation time is correlated with bringing the calculations to a higher dimensional space.

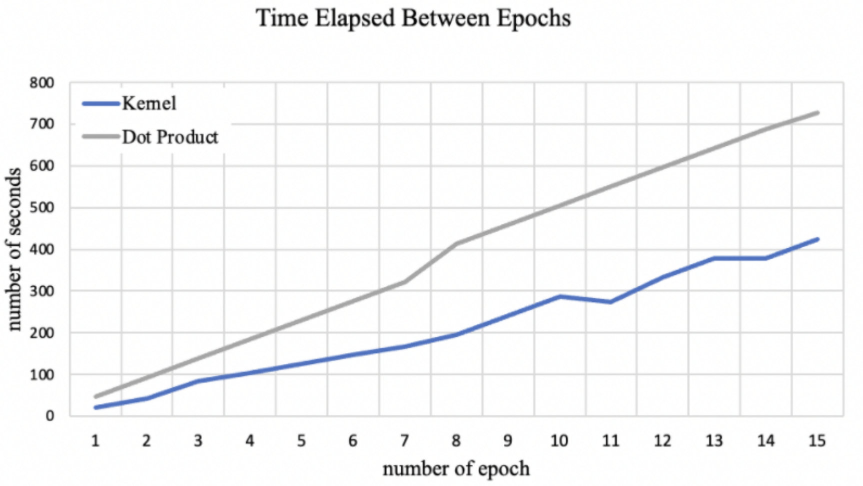
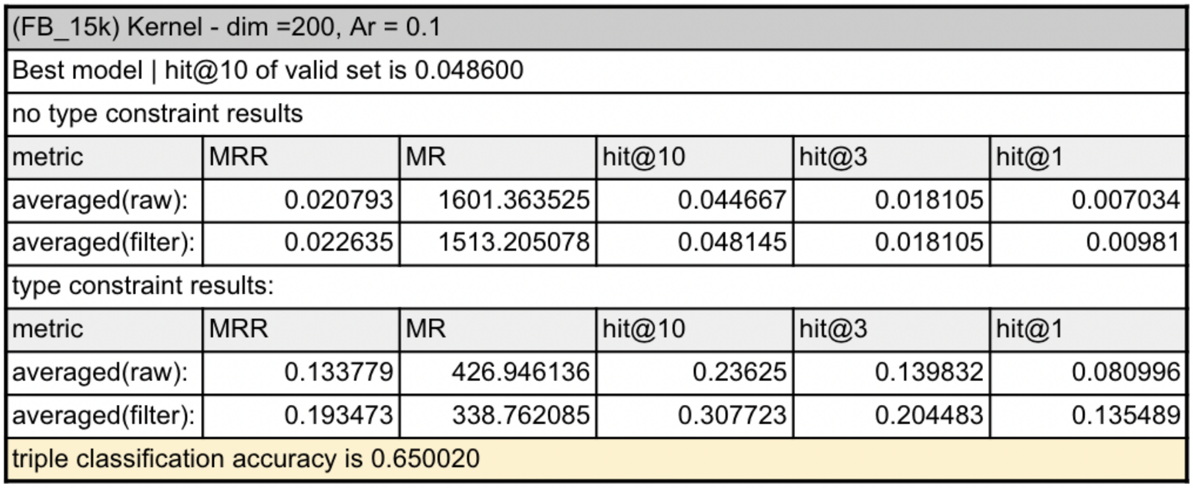


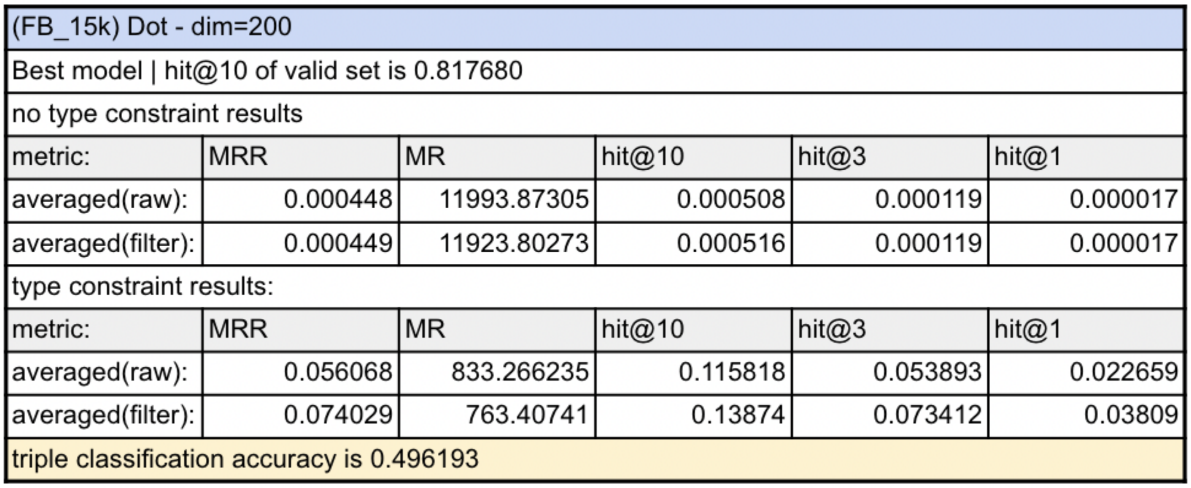
Figure 4: Training speed comparison between Quaternion with Kernel scoring function and Quaternion with dot product scoring function.

In Figure 5 and Figure 6, Hit@n measures the proportion of correct entities in the top n entities. MR measures the average rank of all correct entities with a lower value representing better performance. MRR is the average inverse rank for correct entities. The first observation is that prior knowledge on relation-types (type constraint results) improves the link-prediction results of the implementation using the Quaternion kernel more than its dot product counterpart when the embedding is in dimension 200. This difference in improvements is less significant when the embedding is in dimension 100. This result is not easy to interpret, since the original type-constraint paper stated that improvements are especially prominent when a low model complexity is enforced.



(FB_15k) Kernel - dim =200, Ar = 0.1					
Best model | hit@10 of valid set is 0.048600					
no type constraint results					
metric	MRR	MR	hit@10	hit@3	hit@1
averaged(raw):	0.020793	1601.363525	0.044667	0.018105	0.007034
averaged(filter):	0.022635	1513.205078	0.048145	0.018105	0.00981
type constraint results:					
metric	MRR	MR	hit@10	hit@3	hit@1
averaged(raw):	0.133779	426.946136	0.23625	0.139832	0.080996
averaged(filter):	0.193473	338.762085	0.307723	0.204483	0.135489
triple classification accuracy is 0.650020					

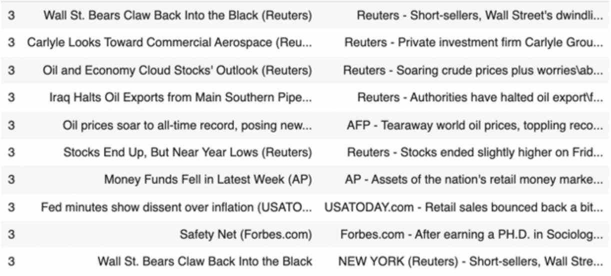
Figure 5: Test result of QuatE with Kernel scoring function, Ar = 0.1, and embedding dimension=200, after training for **50** epochs (approx. 40 minutes)



(FB_15k) Dot - dim=200					
Best model | hit@10 of valid set is 0.817680					
no type constraint results					
metric:	MRR	MR	hit@10	hit@3	hit@1
averaged(raw):	0.000448	11993.87305	0.000508	0.000119	0.000017
averaged(filter):	0.000449	11923.80273	0.000516	0.000119	0.000017
type constraint results:					
metric:	MRR	MR	hit@10	hit@3	hit@1
averaged(raw):	0.056068	833.266235	0.115818	0.053893	0.022659
averaged(filter):	0.074029	763.40741	0.13874	0.073412	0.03809
triple classification accuracy is 0.496193					

Figure 6: Test result of QuatE with dot product scoring function and embedding dimension=200, after training for **50** epochs (approx. 1 hour)

The kernel scoring function surpasses in triple classification accuracy by approximately 23.7%, but it no longer has this advantage in dimension 100 (acc=0.494188 for Quaternion kernel, acc=0.496066 for Quaternion dot product).



3	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...
3	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...
3	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...
3	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...
3	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...
3	Stocks End Up, But Near Year Lows (Reuters)	Reuters - Stocks ended slightly higher on Frid...
3	Money Funds Fell in Latest Week (AP)	AP - Assets of the nation's retail money marke...
3	Fed minutes show dissent over inflation (USATO...	USATODAY.com - Retail sales bounced back a bit...
3	Safety Net (Forbes.com)	Forbes.com - After earning a PH.D. in Sociolog...
3	Wall St. Bears Claw Back Into the Black	NEW YORK (Reuters) - Short-sellers, Wall Stre...

Figure 7: Examples of sentences and labels of the text classification benchmark AGNews.

4.2 Semantic Analysis

AG is a collection of more than 1 million news articles constructed by Xiang Zhang from choosing 4 largest classes (World, Sports, Business, Sci/Tech) from the original corpus. There are 3 columns in each line (Figure 7), corresponding to class index (1 to 4), title and description.For Character Level Embedding, I first perform One-Hot-Encoding based on an alphabet that maps every character and sentential symbol to its numerical representation. For Word Embedding, I leveraged a vector representation of words pre-trained by the GloVe model on a massive 2.2M vocab Common Crawl dataset. Lastly, an LSTM with the same minimum architecture is used to train text classification for each of the three methods of text embedding. The results are shown in Figure 8.

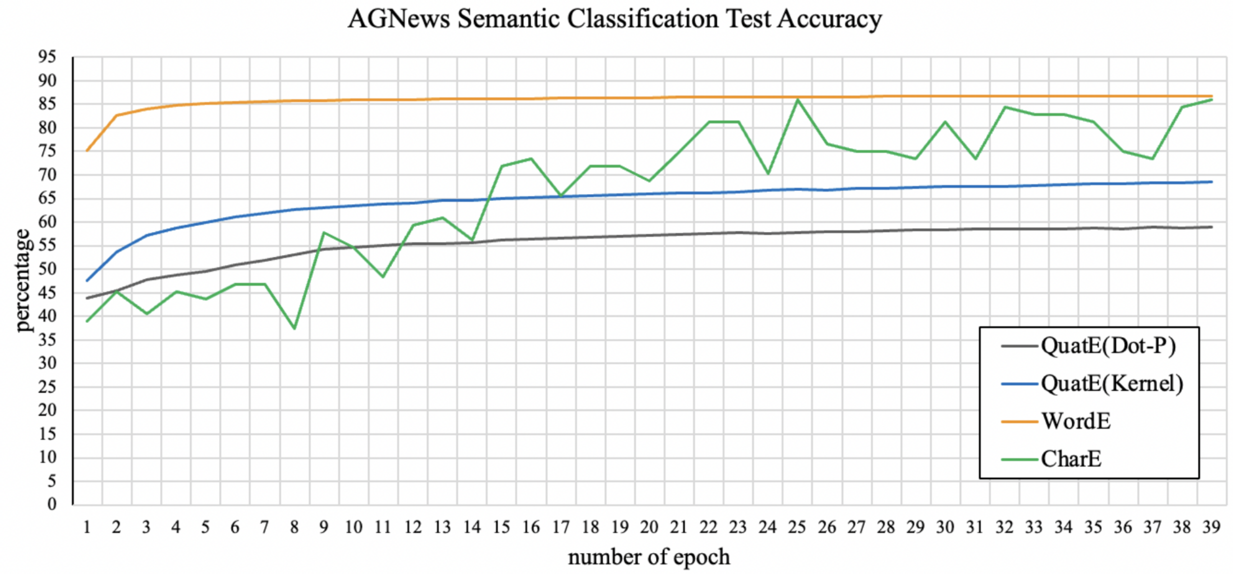


Figure 8: Test accuracy on AGNews in percentage for QuatE (Dot-P), QuatE (Kernel), WordE, CharE.

Overall, word-level embedding achieves the highest accuracy. The performance of character-level embedding tends to oscillate but is comparative to word-level embedding in the end. The Quaternion embedding trained on WNRR18 using Quaternion kernel scoring outperformed the Quaternion dot product scoring. Quaternion embedded vector representation underperforms the other two baselines on this task. A hypothesis is because names, places, and prepositions occur frequently in news articles, but are mostly absent in the entities of WNRR18. The results prove that Quaternion knowledge graph embedding provides an effective way to represent the latent and relational properties in the graph.

6 Citation

[1] S. Zhang, Y. Tay, L.Yao, and Q. Liu. “Quaternion Knowledge Graph Embedding.” CoRR. 2019
[2] S. Choudhary, T. Luthra, A. Mittal, and R. Singh. “A Survey of Knowledge Graph Embedding and Their Applications.” CoRR. 2021.
[3] F. A. Tobar and D. P. Mandic, "Quaternion Reproducing Kernel Hilbert Spaces: Existence and Uniqueness Conditions," in IEEE Transactions on Information Theory, vol. 60, no. 9, pp. 5736-5749, Sept. 2014, doi: 10.1109/TIT.2014.2333734.
[4] Pennington, J. (n.d.). Glove: Global vectors for word representation. Retrieved December 11, 2021, from https://nlp.stanford.edu/projects/glove/.