Aggregating Grouped Semantics as a Building Blocks for Token Representation in Language Models

Anonymous ACL submission

Abstract

Standard language models employ unique, monolithic embeddings for each token, potentially limiting their ability to capture the multifaceted nature of word meanings. We investigate whether tokens can be more effectively represented through a compositional structure that accumulates diverse semantic facets. To explore this, we propose Aggregate Semantic Grouping (ASG), a novel approach leveraging Product Quantization (PQ). We apply ASG to standard transformer architectures (mBERT, XLM-R, mT5) and evaluate this representational scheme across diverse tasks (NLI, NER, QA). Our findings demonstrate that representing tokens compositionally via ASG is able to achieve extreme compression in embedding parameters (0.4-0.5%), while maintaining > 95% task performance relative to the base model, even in generative tasks. These results validate the principle that tokens can be effectively modeled as combinations of shared semantic building blocks. ASG offers a simple yet concrete method for achieving this, showcasing how compositional representations can capture linguistic richness while enabling more compact models.

1 Introduction

In modern language models, each token is typically represented by an individual, unique embedding. However, this approach may not be optimal, as semantically similar tokens (e.g., "mother," "mom," and their respective translations in different languages) can be assigned entirely distinct representations, potentially overlooking shared conceptual underpinnings. Zhang et al. (2024) proposed concept-level representations, grouping semantically similar tokens, using k-means. While this method achieved significant vocabulary compression with retained performance, it struggles with polysemy (e.g., "father" as family vs. religious figure) and is limited to encoder-only models, hindered by not explicitly predicting subword in autoregressive decoding.

To address these limitations, we introduce Aggregate Semantic Grouping (ASG). ASG maintains concept-level sharing but represents tokens as sequences of 'conceptIDs', thereby accumulating multiple semantic facets. This sequence-based representation is inspired by successful applications in information and generative retrieval (Wang et al., 2022; Tay et al., 2022; Zhou et al., 2022). We employ Product Quantization (PQ) (Jégou et al., 2011) to transform tokens into these conceptID sequences, aiming to preserve token's uniqueness and nuances while benefiting from shared semantics.

Our primary contribution is the introduction of Aggregate Semantic Grouping (ASG), a novel method leveraging Product Quantization to represent tokens as sequences of shared conceptIDs, thereby capturing multiple semantic facets while significantly compressing embedding layer parameters. We provide a detailed methodology for applying ASG to both encoder and encoder-decoder transformer models. Conducting experiments across diverse tasks (NLI, NER, QA) and models (mBERT, XLM-R, mT5), we demonstrate that even with extreme compression on embeddings (down to 0.4-0.5% of the original embedding parameters), ASG maintains high performance (often >95% relative to baseline) and outperforms the prior semantic grouping method (Zhang et al., 2024), even in zero-shot cross-lingual transfer scenarios. Our code will be released upon acceptance.

2 Aggregate Semantic Grouping (ASG)

Our approach, Aggregate Semantic Grouping (ASG), reframes token representation by learning compositional embeddings from pre-trained models.

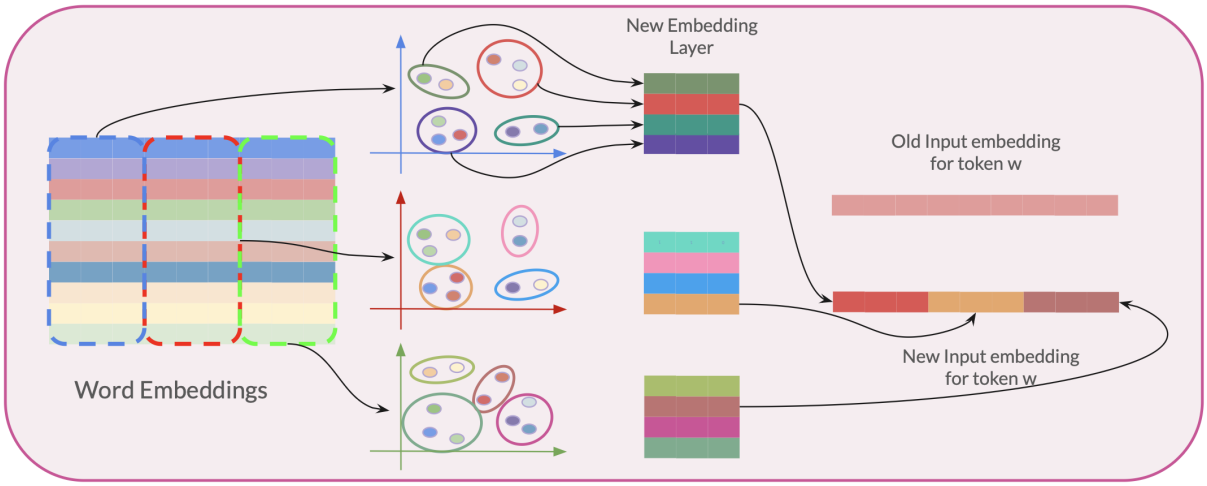

Figure 1: Overview of the Aggregate Semantic Grouping (ASG) method for creating compositional token embeddings. Product Quantization is applied to the original word embedding layer. Embeddings are segmented into m sub-vectors. For each of the m segment positions, k-means clustering is performed on the corresponding sub-vectors from all tokens to learn a codebook of k Concept Vectors (centroids). The new ASG embedding layer containing these learned Concept Vectors is initial as the embedding layer. Instead of using the original input embedding for a token 'w', a sequence of m ConceptIDs used to get their respective Concept Vectors from the ASG layer. These are concatenated to form the new representation for token 'w'.

2.1 Learning Concept Vectors via Product Quantization

We begin with a pre-trained word embedding matrix E, where each row is a D-dimensional vector for a token in a vocabulary of size V. Using Product Quantization (PQ), each D-dimensional embedding is first divided into m distinct segments (sub-vectors), each of dimension D/m. For each of these m segment positions, we apply k-means clustering to the collection of all corresponding segments from every token in the vocabulary. This process yields m distinct codebooks; each codebook C_i (for $i = 0, \ldots, m-1$) contains k centroids, termed **Concept Vectors**, specific to that segment position. Each Concept Vector is of dimension D/m.

2.2 ASG Embedding Layer Initialization

The m distinct codebooks $(C_0, C_1, \ldots, C_{m-1})$, where each codebook C_i contains k Concept Vectors of dimension D/m, are concatenated to form a single, new embedding matrix E'. This matrix E' has dimensions $(m \times k) \times (D/m)$ and stores all unique Concept Vectors. Specifically, the j-th Concept Vector (where $j \in [0, k-1]$) from the i-th codebook C_i is located at row $i \times k + j$ within E'.

Each token is then mapped to a sequence of m **ConceptIDs**. For each of its m embedding segments, the corresponding ConceptID is the specific row index in E' that stores the chosen Concept Vector for that segment. This row index is determined as $i \times k + s_i$, where i is the segment index (from 0 to $m-1$) and s_i is the index (from 0 to $k-1$) of the selected centroid from the i-th segment's codebook. This sequence of m row indices (ConceptIDs) thus identifies the set of Concept Vectors representing the token.

2.3 Token Representation with ASG

When a token is processed, its pre-computed sequence of m ConceptIDs is used to retrieve the corresponding m Concept Vectors from their respective codebooks within E'. Let these retrieved Concept Vectors be $v_0, v_1, \ldots, v_{m-1}$, where each v_i has dimension D/m. The final ASG representation for the token, $e' \in \mathbb{R}^D$, is obtained by concatenating these m Concept Vectors:

$$e' = \text{concat}(v_0, v_1, \ldots, v_{m-1}) \quad (1)$$

This vector e' serves as the input to subsequent layers of the model.

2.4 Application to Generative Models

For model with decoder, which have separate input and output embedding layers (the latter often serving as token classifier weights), we apply the ASG process to both. This results in two distinct ASG embedding structures: one for input token representations (E') and another for the output layer

Table 1: Evaluation results across cluster granularities for MBERT and XLM-R on multilingual benchmarks. Scores include F1, Accuracy, and relative performance (%Base). For XNLI %Base is for the accuracy relative to the base model. 40% SG: Semantic Grouping as mentioned in Zhang et al. (2024). In the Zero-Shot setting the models were trained on english dataset and have been tested on all the languages.

Model	Parameter Reduced to (%)		XNLI			WikiANN		Zero-Shot			
								XNLI		WikiANN	
	Embedding	Model	Accuracy	F1	%Base	F1	%Base	Accuracy	%Base	F1	%Base
MBERT	100.00	100.00	75.46	74.79	100.00	89.74	100.00	64.86	100.00	58.58	100.00
-40% SG	40.00	68.95	72.43	71.88	95.99	86.69	96.61	60.64	93.49	52.35	89.37
-ASG(k=512, m=48)	0.50	48.65	73.51	72.84	97.42	88.11	98.19	61.30	94.51	55.71	95.10
XLM-R	100.00	100.00	77.98	77.28	100.00	88.37	100.000	71.94	100.00	58.74	100.00
-40% SG	40.00	58.48	74.56	73.96	95.61	84.57	95.70	65.83	91.51	51.48	87.65
-ASG(k=1024, m=48)	0.40	31.08	77.06	76.39	98.81	86.53	97.92	68.05	67.39	54.46	92.72

(OE'), each derived from their respective original embedding matrices.

Output Logit Calculation: To compute the logit l_t for a target token t, the final hidden state $H \in \mathbb{R}^D$ from the model is first segmented into m parts: $H = [H_0, H_1, \ldots, H_{m-1}]$, where each $H_i \in \mathbb{R}^{D/m}$. Let the sequence of Concept Vectors for token t be $u_{t,0}, u_{t,1}, \ldots, u_{t,m-1} \in OE'$. The logit is calculated as:

$$l_t = \sum_{i=0}^{m-1} H_i \cdot u_{t,i} \quad (2)$$

3 Experiments and Results

3.1 Datasets

We evaluate our proposed ASG method on diverse cross-lingual benchmarks for natural language inference (NLI), question answering (QA), and named entity recognition (NER). These include: **XNLI** (Conneau et al., 2018), a 15-language sentence understanding benchmark; the Gold Passage (GoldP) task of **TyDi QA** (Clark et al., 2020), an 11-language QA dataset where gold context is provided; and the XTREME benchmark version (Hu et al., 2020) of **WikiANN** (Pan et al., 2017), a 40-language NER dataset.

For k, values were generally chosen as powers of two. This allowed us to systematically target specific levels of embedding parameter compression, aiming for reductions that brought the ASG embedding layer size to approximately 0.5%, 1%, and 4% of the original embedding parameters. Regarding the number of subspaces m, our explorations indicated that too few subspaces (e.g., $m = 16$) resulted in a significant degradation of model performance. Conversely, using very high values for m (e.g., 128, 256, or 512), would lead to extremely small dimensions for each segment (D/m, potentially as low as 4, 2, or 1 for common embedding sizes D) and would consequently require very long sequences of ConceptIDs (length m) to represent each token. These considerations led us to focus on m values within a moderate range for the experiments detailed below.

3.2 Fine-tuning Performance

We evaluated ASG on encoder-only mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2019) models using the XNLI and WikiANN datasets, mainly to compare it's effectiveness against the Semantic grouping as mention in Zhang et al. (2024). As demonstrated in Table 1, ASG achieves significant embedding compression while maintaining over 97% of baseline performance and notably outperforms Semantic Grouping (SG) method, even with a low k value.

To assess ASG for generative tasks, we then evaluated the mT5 model (Xue et al., 2020) on the TyDiQA and WikiANN datasets, applying ASG to both its input and output embeddings. Table 2, detailing results for various cluster (k) and subspace (m) configurations, shows ASG consistently achieved over 85% of baseline mT5 performance. Specifically, with $k \geq 2048$, relative performance on TyDiQA surpassed 90%, while on WikiANN, ASG configurations generally exceeded 95% of the baseline.

Furthermore, for mT5, we investigated a variant employing a single shared codebook across all m subspaces. To achieve this, the m segments from all token embeddings in the vocabulary are pooled together before applying k-means clustering. This yields one global codebook of Concept Vectors. Each of the m ConceptIDs for a token then selects a Concept Vector from this single

Table 2: Evaluation results for Generative models across cluster granularities for MT5 on TyDiQA and WikiANN. Seperate: 1 codebook per segment, Shared: codebooks shared across all segments, In the Zero-Shot setting the models were trained on English dataset and have been tested on all the languages.

	Model	Parameter Reduced to (%)		TyDiQA			WikiANN		WikiANN (Zero-Shot)	
		Embedding	Model	F1	EM	%Base	F1	%Base	F1	%Base
	MT5	100.00	100.00	70.74	56.20	100.00	84.21	100.00	50.75	100.00
ASG Separate	-($k = 1024, m = 32$)	0.45	15.06	60.67	46.15	85.76	79.85	94.82	25.84	50.91
	-($k = 2048, m = 32$)	0.85	15.41	63.81	49.06	90.19	80.93	96.11	29.85	58.82
	-($k = 8192, m = 32$)	3.32	17.51	66.22	51.71	93.61	82.19	97.60	33.51	66.03
	-($k = 1024, m = 64$)	0.45	15.06	69.96	55.53	98.89	83.18	98.78	44.02	86.74
ASG Shared	-($k = 16384, m = 32$)	0.25	14.89	66.50	51.90	93.99	81.65	96.96	34.01	67.02
	-($k = 32768, m = 32$)	0.45	15.06	67.00	53.06	94.71	82.04	97.42	37.01	72.92
	-($k = 32768, m = 64$)	0.25	14.89	70.81	56.51	100.09	84.19	99.97	47.23	93.06

shared codebook to represent its corresponding segment. This shared codebook is then used across all m positions for constructing the token representation. This approach, despite reducing the diversity of available Concept Vectors, impressively maintained over 95% relative performance across both TyDiQA and WikiANN. This suggests that a highly restricted set of output Concept Vectors can still be effective for generative tasks.

3.3 Cross-Lingual Transfer (Zero-Shot)

For zero-shot cross-lingual transfer, we followed the experimental setup of Zhang et al. (2024). Models were trained solely on the English XNLI and WikiANN training sets and then evaluated on the multilingual test sets of these datasets. In this setting, ASG-enhanced models outperformed the Semantic Grouping method. While generative models using ASG with lower k (clusters per segment) and m (segments) values showed reduced performance in cross-lingual transfer (Table 2), configurations with $m = 64$ segments nonetheless achieved at least 86% relative to baseline model performance. Using shared codebook, the performance further improved upto 93% relative to the baseline model, with just 0.25% of the embedding parameters.

3.4 Qualitative Analysis

Figure 2 illustrates how Aggregate Semantic Grouping (ASG) captures varied semantic facets of the token "father" through its clustering across selected segments:

- **Familial Context:** "father" clusters with kinship terms such as "padre" (father), "mother", and "daughter" (Segment 2), or "barn" (child), "parent", and "grandmother" (Segment 12; also Segment 16), reflecting its primary familial sense.
- **Authority/Religious Context:** In Segment 0, "father" groups with "Chief", "Prophet", "notables", and "religión", indicating connotations of leadership or religious reverence.
- **Figurative/Abstract Contexts:** Other segments link "father" to broader concepts, such as "Zeus" (mythological father figure), or with terms like "records", "govern", and "legacy" (Segment 7), potentially reflecting historical origin, or the act of establishing something significant.

4 Conclusion

This work investigated equipping language models with shared, compositional token representations as an alternative to traditional monolithic embeddings. We explored this through Aggregate Semantic Grouping (ASG), where Product Quantization transforms embeddings into sequences of **ConceptIDs** that map to shared, learned **Concept Vectors**, enabling multifaceted semantic capture alongside significant compression. Extensive experiments on diverse models (including mBERT, XLM-R, and mT5) and NLU tasks (such as NLI, NER, and QA) found ASG maintains high performance (often >95% relative to baseline) despite extreme parameter reduction (to <1% of original size). ASG also outperformed prior semantic grouping methods, and proved effective for generative architectures. These findings confirm that ASG's decomposition of tokens into shared components offers an efficient, semantically rich, and promising direction for language modeling; future work may explore dynamic or adaptive quantization techniques.

5 Limitations

ASG was applied directly to word embeddings from pre-trained models without an explicit cross-

lingual alignment step, which could refine Concept Vector clustering. This may partly explain the observed performance degradation in generative tasks within cross-lingual settings, such as on WikiANN, where better nuance preservation through alignment-optimized clustering could be beneficial. Furthermore, we did not undertake continual pre-training of the models with the ASG embeddings; such a phase could allow models to more effectively adapt to the compositional representations and potentially enhance overall performance.

References

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *International conference on machine learning*, pages 4411–4421. PMLR.

Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, and 1 others. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, and 1 others. 2022. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems*, 35:25600–25614.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Xinyu Zhang, Jing Lu, Vinh Q Tran, Tal Schuster, Donald Metzler, and Jimmy Lin. 2024. Tomato, tomahto, tomate: Measuring the role of shared semantics among subwords in multilingual language models. *arXiv preprint arXiv:2411.04530*.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An ultimate retriever on corpus with a model-based indexer. *arXiv preprint arXiv:2208.09257*.

A Experimental Setup

All our experiments are conducted using the smallest available checkpoint for each respective pretrained model. Training is performed with a batch size of 128, and all experiments were run on a single Nvidia L40 GPU.

For the encoder models (mBERT and XLM-R), we set a weight decay of 0.01. The learning rate was 5×10^{-6} for XNLI experiments and 5×10^{-5} for WikiANN experiments. These models were trained for 2 epochs; for cross-lingual transfer settings, training was extended to 5 epochs. The mT5 model was trained with a learning rate of 1×10^{-3}.

Product Quantization is implemented using the `nanopq` library[1]. For the k-means clustering within `nanopq` we use the `faiss` library[2].

A.1 ASG Configuration and Embedding Parameters

Table 3 provides a summary of the configurations used for Aggregate Semantic Grouping (ASG) across different models, alongside details for the original base models. The table specifies the choices for k (number of centroids per subspace) and m (number of subspaces) for each ASG setup. It also lists the resulting total number of parameters in the model and the shape of the embbeding layer.

[1] https://github.com/matsui528/nanopq

[2] https://faiss.ai/

Table 3: Model Configurations and Embedding Parameters for ASG, Underlined uses a shard codebook

Model	**k**	**m**	**Parameters**	**Embedding Shape (Dim)**
mBERT	N/A	N/A	177M	[120k, 768]
	512	48	86M	[30k, 16]
XLM-R	N/A	N/A	277M	[250k, 768]
	1024	48	86	[49k, 16]
mT5	N/A	N/A	300M	[256k, 512]
	1024	32	45M	[36k, 16]
	2048	32	46M	[68k, 16]
	8196	32	53M	[265k, 16]
	16392	32	44M	[20k, 16]
	32784	32	45M	[36k, 16]
	32784	64	44M	[39k, 8]
	1024	64	45M	[72k, 8]

Segment (m=48)	Number of tokens in Cluster	Words in Cluster
0	27	笔', '筆', '钦', '할', '**father**', 'parts', 'Chief', " "'chief', 'Ho', 'ābinädons', '**Mother**', 'olid', '##öld', 'Telegraph', 'earth', " "'##ვან', 'notables', 'adta', '**religión**', 'метал', 'İsrail', '**praise**', " "'**Prophet**', 'שירת', 'Thief', 'voter', 'Capitaine'
1	29	丑', 'father', 'piccolo', 'Zeus', 'Castello', " "'центру', '##ام', 'senjata', '##ơi', 'antichi', 'iniziale', 'جسم', " "'Verenigd', 'apariencia'
2	22	**father**', '**padre**', '**daughter**', '**mother**', 'сын', " "'ते', 'incidente', '**baby**', '**daughters**', 'V6', '**grandson**', 'नया॰हाल', " "'afromoths', 'Michaela', 'משפחתו', 'семейството', 'Мала', 'lluz', '운영', " "'Potok', 'смт', '##길'
7	19	曜', 'top', 'father', 'records', 'govern', " "'##jnë', 'corona', 'Hartmann', '##jска', 'ՍիջազգայինՆ', 'Dakar', 'Pons', " "'legacy', '##нскі', 'портрет', 'marge', 'naturally', 'आंतरराष्ट्रीय', " "'McDowell
12	24	##ز', 'father', 'él', 'barn', '##بی', 'بزرگ', " "'Infantry', 'Elementary', 'kinderen', 'coupe', 'parent', '##CA', 'injuries', " "'##RC', '##GC', 'connect', 'cache', 'قطع', 'kaldı', 'Lahti', 'indicato', " "'seco', 'grandmother', 'wheat'
16	13	father', 'début', 'mother', 'port', 'складі', " "'Dal', 'Beginning', 'uncle', '##дним', 'Straży', 'começament', '##rusade', " "'grandmother'

Figure 2: Grouping of the token "father" at a few selected subspaces