Unsupervised Dialogue Topic Segmentation in Hyperdimensional Space

Seongmin Park\textsuperscript{1}, Jinkyu Seo\textsuperscript{2}, Jihwa Lee\textsuperscript{1}

\textsuperscript{1}ActionPower, Seoul, Republic of Korea
\textsuperscript{2}Dept. of Applied Biology and Chemistry, Seoul National University, Seoul, Republic of Korea

seongmin.park@actionpower.kr, seojk413@snu.ac.kr, jihwa.lee@actionpower.kr

Abstract

We present HyperSeg, a hyperdimensional computing (HDC) approach to unsupervised dialogue topic segmentation. HDC is a class of vector symbolic architectures that leverages the probabilistic orthogonality of randomly drawn vectors at extremely high dimensions (typically over 10,000). HDC generates rich token representations through its low-cost initialization of many unrelated vectors. This is especially beneficial in topic segmentation, which often operates as a resource-constrained preprocessing step for downstream transcript understanding tasks. HyperSeg outperforms the current state-of-the-art in 4 out of 5 segmentation benchmarks – even when baselines are given partial access to the ground truth – and is 10 times faster on average. We show that HyperSeg also improves downstream summarization accuracy. With HyperSeg, we demonstrate the viability of HDC in a major language task. We open-source HyperSeg to provide a strong baseline for unsupervised topic segmentation.

Index Terms: topic segmentation, hyperdimensional computing, summarization

1. Introduction

Automatic speech recognition (ASR) is often paired with downstream natural language processing (NLP) tasks for information extraction. Appropriately segmenting ASR transcripts to conform to the limited input length of downstream models is a practical and important consideration for such hybrid systems. Conventional NLP models cannot process whole meeting transcripts (which typically include hundreds or thousands of utterances spanning several hours) in a single inference context. Such physical constraints necessitate effective methods to divide transcripts into coherent segments before passing them on to downstream models.

Much research has focused on unsupervised approaches for topic segmentation due to the scarcity of labeled datasets with diverse domain and lexical coverage. Because segmentation is performed at the utterance level, dominant approaches to unsupervised topic segmentation use neural utterance embeddings. We find two problems with this line of research: neural utterance embeddings are very brittle to the domain in question \cite{1, 2}, and existing unsupervised topic segmentation methods cannot determine segment counts on their own. Even state-of-the-art (SOTA) systems rely on carefully tuned hyperparameters, without which the systems offer lower segmentation accuracy than random segmentation.

This work exploits hyperdimensional computing (HDC) to create a robust topic segmentation system resilient to domain and hyperparameter change. HDC leverages vector interactions in high-dimensional spaces to create scalable, aggregated representations. We initialize each token in an utterance as a random 10,000-dimensional vector and bind those token representations to portray a complete utterance. HDC’s aptitude for single-pass learning makes it inherently robust to domain shift \cite{3}. Compared to neural approaches, HDC produces more representative utterance embeddings in a fraction of the processing time (Figure 1). We summarize our contributions as follows:

1. We introduce HyperSeg, the first framework to use HDC for topic segmentation.
2. HyperSeg outperforms the best unsupervised transcript segmentation baselines across multiple domains, even when baselines are provided with optimal hyperparameters and partial ground truth.
3. HyperSeg is an order of magnitude faster than neural baselines, at an average of 10 times per utterance. HyperSeg achieves such speedup while running entirely on the CPU.
4. HyperSeg increases downstream summarization accuracy compared to baseline and naive segmentation schemes.

2. Related work

2.1. Hyperdimensional computing

The concentration of measure \cite{7} is a phenomenon that occurs among a set of randomly initialized \(n\)-dimensional vectors where – given sufficiently large \(n\) – the cosine similarity between each vector is heavily concentrated around 0. Such a...
set of vectors are said to be pseudo-orthogonal to each other. In other words, a random vector drawn in high-enough dimensions will be probabilistically dissimilar to those previously drawn. Hyperdimensional computing (HDC) [8] is a class of vector symbolic architectures that leverages such convenient orthogonality in high-dimensional space. HDC designates a random large dimensional vector (with $n \geq 10,000$) to represent concepts of varying granularity, such as a single text token, a sentence, or an entity class.

HDC occupies the middle ground between symbolic artificial intelligence (e.g. agent-based modeling) and distributional artificial intelligence (e.g. neural networks), amending the shortcomings of both. In symbolic architectures, one-hot entity representations render entity similarity calculations discontinuous and thus non-interpolable [9, 10]. At the opposite end of the spectrum, distributional semantics uses vector representations for natural interpolation and loss propagation, but cannot consolidate learned vectors into order- and dimension-preserving sequences (known as the “binding problem” [11]). HDC retains the benefits of distributed vector representations while offering a solution to the binding problem by tapping into the aforementioned pseudo-orthogonality of vectors at high dimensions. At extremely high vector spaces, the pseudo-orthogonality of randomly realized vectors converges to exact orthogonality. The architecture also comes with the benefit of extremely fast and parallelizable representation construction.

HDC has seen adoption in time-sensitive real-time systems [9], often in conjunction with artificial neural networks. The framework has found its place in text processing as well [12], boasting high accuracy and extremely fast processing time. For the first time, we apply HDC’s representational power to transcript segmentation. We find that utterance embeddings created with HDC are semantically more coherent than ones created with the best sentence embedding neural networks.

2.2. Unsupervised topic segmentation

SOTA unsupervised topic segmentation algorithms first construct vector representations of each transcript utterance, to be delivered to downstream sequence segmentation algorithms. BERTSeg [13] and CohereSeg [14] use SentenceBERT [5] embeddings to represent each utterance. BERTSeg uses off-the-shelf pre-trained BERT [15] models, while CohereSeg further trains BERT with utterance-pair coherence loss. Semantic utterance vectors are fed to the TextTiling algorithm [16], which is a linear sequence segmentation scheme that determines topic segments by vector distance. GraphSeg [17] uses an alternative segmentation scheme by constructing a semantic graph with each utterance as a node. Utterance communities are determined in the semantic graph, and maximal cliques in the graph are designated as transcript segments.

Depending on the transcript domain, however, the representational capability of semantic embeddings can falter. Each utterance embedding contains an unexplainable juxtaposition of semantic and lexical concepts [1, 18]. We find existing transcript segmentation algorithms are brittle in both representation construction and downstream segmentation. Such brittleness makes SOTA baselines extremely sensitive to hyperparameters, often resulting in lower accuracy than random segmentation. We show that replacing the utterance embedding step with HDC produces more topic-aware utterance vectors across multiple datasets. Leveraging such representational advantages, HyperSeg eliminates all hyperparameters while outperforming previous SOTA, even when optimal hyperparameters (including partial ground truth, such as the total segment count of a transcript) are provided to baselines. HyperSeg also operates an order of magnitude faster than neural baselines, which are often unfit for resource-constrained preprocessing tasks such as topic segmentation.

### Table 1: Summary of benchmark datasets. “Uts” and “Utt len” stand for mean utterance count and mean utterance length.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Uts</th>
<th>Utt len</th>
<th>Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMI</td>
<td>Meeting</td>
<td>364.10</td>
<td>36.88</td>
<td>6.82</td>
</tr>
<tr>
<td>ICSI</td>
<td>Meeting</td>
<td>879.51</td>
<td>50.84</td>
<td>16.12</td>
</tr>
<tr>
<td>Doc2Dial</td>
<td>Customer service</td>
<td>30.13</td>
<td>77.57</td>
<td>1.42</td>
</tr>
<tr>
<td>VT-SSum</td>
<td>Lecture</td>
<td>306.10</td>
<td>82.72</td>
<td>33.87</td>
</tr>
<tr>
<td>Wiki-727k</td>
<td>General</td>
<td>46.43</td>
<td>125.36</td>
<td>4.66</td>
</tr>
</tbody>
</table>

### 3. Segmentation with HyperSeg

#### 3.1. Constructing utterance embeddings

We follow [12] and create utterance embeddings by binding multiple hyperdimensional word embeddings. For every word in an transcript that is not a stopword, we randomly initialize a 10,000-dimensional bipolar vector $w$:

$$w = (x_0, \ldots, x_9, 999),$$

where $x \sim \text{IID}\{-1, 1\}$.

Each element in the word embedding is either -1 or 1, and same words are mapped to the same embedding. High dimensionality ensures that each word embedding is roughly orthogonal to every other word embedding (“blessing of dimensionality” [19]).

To represent a single utterance, we use permutation binding [9, 12] to consolidate all word embeddings in the utterance into a single hyperdimensional vector, also of dimension 10,000. To generate an utterance vector $u$, we first encode the position of each constituent word vector $w_i$ by right-shifting ($l - i$) times. $l$ is the utterance word count and $i$ is the zero-based word index within $u$. The final utterance embedding is the component-wise majority vote ($M$) of all of its position-
encoded word vectors ($w_{i}^{pos}$):
\[ w_{i}^{pos} = \prod_{i}^{j-1} (w_{i}) \]  \hspace{1cm} (2)
\[ u = M(w_{0}^{pos}, w_{1}^{pos}, ..., w_{N}^{pos}) \]  \hspace{1cm} (3)

Topic segmentation does not require extracting constituent word embeddings after obtaining utterance embeddings. Even so, the reversibility of right shifts and position-wise multiplication guarantees the ability to recover each word embedding should the need arise.

3.2. Transcript segmentation

We calculate the boundary similarity score ($S$) for every neighboring utterance pair $w_i$ and $u_{i+1}$:
\[ S = \frac{u_i \cdot u_{i+1}}{|u_i| |u_{i+1}|} \]  \hspace{1cm} (4)

Each boundary score is the cosine similarity between surrounding utterance embeddings. An utterance boundary index $j$ has to satisfy two conditions to be selected as a topic boundary: its similarity score $s_j$ has to be a local minimum, and $s_j$ has to be better than an automatically determined threshold $T$.

Since the desired number of total segments is unknown, selecting a similarity threshold for topic boundaries is a non-trivial process. Baselines rely on hyperparameters to determine the total segment count, which often hinders segmentation accuracy at test time. HyperSeg automatically calculates the similarity cutoff threshold $T$ as one standard deviation of boundary scores ($\sigma_s$) subtracted from the mean ($\mu_s$). We design $T$ in such a way to incorporate both local (cosine distance between neighboring embeddings) and global (mean and standard deviation of all distances) statistics. The number of topics in a document is therefore adaptively determined.

Local minima boundary indices from $S$’s index set $I(S)$ with scores below $T$ are selected as the set of final segment boundaries $B$ (Figure 2):
\[ T = \mu_s - \sigma_s \]  \hspace{1cm} (5)
\[ B = \{ j \in I(S) | s_j \text{ is a local minimum} \land s_j < T \} \]  \hspace{1cm} (6)

HyperSeg can be optionally operated in damp mode, where the number of segments is capped with a logarithm filter. Of utterance boundaries with similarity smaller than $T$, we select $N$ boundaries with the smallest boundary cosine similarity, where
\[ N = \lfloor \log^2 \{ |s | \text{ scores} < T \} \rfloor \]  \hspace{1cm} (7)

Dampened HyperSeg creates segments less eagerly.

4. Experiments

We measure HyperSeg’s effectiveness with two approaches: raw segmentation performance (accuracy and speed) and improvements in downstream summarization of segmented text. We also compare simple word- and n-gram counting with HyperSeg to empirically illustrate how HyperSeg performs beyond naive surface-level lexical matching. We use GraphSeg [17], BertSeg [13], and CohereSeg [14] as state-of-the-art baselines.

4.1. Datasets and metrics

Segmentation accuracy of HyperSeg and baselines are tested on five benchmarks: AMI [20], ICSI [21], Doc2Dial [22], VT-SSum [23], and Wiki-727k [24]. All datasets except Wiki-727k consist of dialogue transcripts. We include Wiki-272k to demonstrate that HyperSeg’s robustness extends to written text segmentation. Selected benchmarks cover diverse domains, including meetings, customer service, lectures, and others. Table 1 includes detailed statistics of each dataset.

Linear segmentation is evaluated as a binary classification problem for each possible utterance boundary. Segmentation accuracy is measured with classification F1 scores and Boundary Similarity [25]. The F1 score measures the harmonic mean between precision and recall for the binary classification problem (boundary-or-not). Boundary Similarity measures the inter-coder agreement between system and reference segmentations.

Downstream summarization accuracy is measured by how much the system summary overlaps with the ground-truth summary. We measure ROUGE-1 ($R_1$), ROUGE-2 ($R_2$), and ROUGE-L ($RL$) [26] between reference and system summaries. An off-the-shelf BART [27] model trained on the standard CNN/Daily Mail dataset [28] is used as the downstream summarizer.

5. Results

5.1. Transcript topic segmentation

HyperSeg outperforms baselines in all datasets except ICSI even when the compared systems use the most optimal hyperparameters (Table 2). For GraphSeg, BertSeg, and CohereSeg, we report their best possible scores by providing the true number of ground-truth segments as a hyperparameter. Only HyperSeg and CohereSeg consistently outperform random segmentation. Performance of all systems noticeably degrades in AMI and ICSI, where true segment boundaries constitute less than 2% of all possible breakpoints. For such sparse datasets where metrics fail to take near-misses into account, we also report downstream summarization performance in the next section. All reported results are averages after 5 runs with different random seeds.
Table 3: Average milliseconds elapsed per utterance. HyperSeg is 10 times faster than the next fastest baseline.

<table>
<thead>
<tr>
<th>Model</th>
<th>AMI</th>
<th>ICSI</th>
<th>Doc2Dial</th>
<th>VT-SSum</th>
<th>Wiki-727k</th>
<th>Mean</th>
<th>× HyperSeg</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphSeg [17]</td>
<td>11.60</td>
<td>13.71</td>
<td>10.66</td>
<td>11.11</td>
<td>9.73</td>
<td>11.36</td>
<td>× 10.42</td>
</tr>
<tr>
<td>BERTSeg [13]</td>
<td>11.92</td>
<td>12.02</td>
<td>12.82</td>
<td>11.45</td>
<td>11.22</td>
<td>11.89</td>
<td>× 10.91</td>
</tr>
<tr>
<td>HyperSeg (ours)</td>
<td>0.77</td>
<td>0.70</td>
<td>1.19</td>
<td>1.28</td>
<td>1.49</td>
<td>1.09</td>
<td>× 1</td>
</tr>
</tbody>
</table>

On average, HyperSeg is 10 times faster per utterance than the next quickest baseline GraphSeg (Table 3). Reported segmentation speeds are measured on an AMD EPYC 7742 CPU and four 80GB A100 GPUs. All baseline runs were conducted on GPUs, while HyperSeg ran entirely on the CPU.

5.2. Downstream summarization performance

To demonstrate the effect of accurate topic segmentation on downstream summarization, we apply HyperSeg to AMI and ISCI before summarizing the transcripts. Compared to no segmentation, naive segmentation of equal segment length, and random segmentation, HyperSeg always improves summarization in terms of ROUGE scores (Table 4).

Table 4: Downstream summarization accuracy. Only HyperSeg yields consistently better ROUGE scores than random or naive baselines.

<table>
<thead>
<tr>
<th>Model</th>
<th>AMI</th>
<th>ICSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>0.19</td>
<td>0.03</td>
</tr>
<tr>
<td>Random</td>
<td>0.27</td>
<td>0.13</td>
</tr>
<tr>
<td>Uniform</td>
<td>0.13</td>
<td>0.08</td>
</tr>
<tr>
<td>GraphSeg [17]</td>
<td>0.22</td>
<td>0.13</td>
</tr>
<tr>
<td>BERTSeg [13]</td>
<td>0.22</td>
<td>0.14</td>
</tr>
<tr>
<td>CohereSeg [14]</td>
<td>0.27</td>
<td>0.06</td>
</tr>
<tr>
<td>HyperSeg (ours)</td>
<td>0.32</td>
<td>0.06</td>
</tr>
</tbody>
</table>

Figure 3: Distribution of utterance vector similarities (VT-SSum). Regardless of tokenization scheme (word-level or n-gram), HyperSeg produces more gradual utterance similarity vectors compared to word or n-gram counting that use one-hot token vectors.

5.3. Difference to simple lexical matching

HyperSeg offers two pronounced advantages over simple word or n-gram count-based topic segmentation: fine-grained sentence similarity scores and flexibility in token granularity. The distributed nature of HyperSeg produces a continuous spread of utterance boundary similarities (Figure 3). Word or n-gram counting yields a sparse spread of utterance similarities. Such segmentation renders the vectors unusable as features in possible downstream classification networks. HyperSeg can also accommodate character-level tokenization by modifying its binding algorithm. Such change in granularity offers a trade-off between robustness against character error rate and representational burden (an utterance vector has to contain more token vectors within a designated dimension compared to word-level tokenization).

6. Conclusion

HyperSeg is a new state-of-the-art topic segmentation algorithm. We find that hyperdimensional computing’s representational benefits can be effectively realized in dialogue topic segmentation. HyperSeg is more accurate and lightweight compared to best-performing baselines. We attribute HyperSeg’s performance to the robustness of HDC embeddings across domains. Such advantages make HyperSeg an ideal component in ASR post-processing, which is often resource-constrained. We confirm that HyperSeg’s representational capabilities extend to downstream summarization as well.

While this paper deals solely with linear transcript segmentation, a more natural approach to dialogue segmentation would take repetitive alternations of topics into consideration. Such methods require gathering related utterances into non-linear topical clusters. Equipped with the representational robustness of hyperdimensional vectors, HyperSeg can be naturally extended to accommodate such advanced schemes for segmentation. We also leave experiments with different utterance tokenization schemes (character-level, subword-level) for future research.

7. References

