THE SURPRISING EFFECTIVENESS OF EXPLICIT SEMANTIC ANALYSIS IN DATALESS CLASSIFICATION

BY

SHASHANK GUPTA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Adjunct Professor Dan Roth

ABSTRACT

Organizing textual content into broad labels is one of the most basic tasks that some people carry out on a regular basis. This simple task helps people navigate through large document collections by exposing the labels of the documents, which can then be used for selecting the documents of interest. Currently, the most popular techniques for providing this basic functionality are supervised in nature, wherein someone has to annotate a collection of documents with the labels of interest. However, it might not always be possible to create a sizeable labeled dataset for every scenario or domain of interest. Thus, techniques like "Dataless Classification" have been proposed in the past that are able to bootstrap the creation of a classifier by only requiring semantic descriptions of the labels. However, despite the encouraging performance of Dataless Classification on Text Classification tasks, there is still a room for large improvement. In this thesis, we identify the limitations of ESA-driven Dataless Classification and systematically design techniques for addressing each limitation. In the process, we end up developing 4 new embeddings – **EntityESA**, **Entity2Vec**, **Topic2Vec** and **Word2Concept**. However, despite our best efforts, we found it difficult to outperform the original Dataless Classification system. For some of the techniques we provide an explanation for this observed behavior, however we also attribute some of these observations to the datasets that are being used for evaluation purposes. We then propose a way to create a new dataset that can used for future Dataless evaluations. The new embedding methods proposed in this work are generic enough that they can be of independent interest as well.

To my family, friends, and mentors for their love, support, guidance and unwavering confidence in my abilities.

ACKNOWLEDGMENTS

First and foremost, I would like to express my heartfelt gratitude to my adviser Professor Dan Roth for giving me the rare opportunity to work with him, and for making me a part of his esteemed research group. I had the utmost privilege of working with him right from my first day in the grad school. Pursuing a graduate degree at one of best schools was a dream that I had been chasing for many years, and without Dan's acceptance and funding, it would have remained a dream. I grew immensely both as a researcher and a person under Dan's tutelage. Not only did Dan help me develop strong foundations in ML and NLP, but he also taught me how to critique the work of others. Dan always showed immense confidence in my abilities, and never treated me differently from his Ph.D. students – I'm indebted to him for that attitude. Dan also helped me develop an appreciation for good software engineering skills, and taught me the importance of making one's research accessible by developing and releasing robust systems. Outside of research as well, Dan gave me ample opportunities to grow. He not only gave me the opportunity to be a Teaching Assistant (TA) for his course on Machine Learning, but also gave me ample flexibility in conducting my discussion sessions and implementing new ideas for improvement. My experience as a Teaching Assistant was really transforming. Not only did that opportunity help me develop immense confidence that will last the entire lifetime, but it also helped me develop a taste and appreciation for teaching and leadership. Furthermore, the TA opportunity helped me meet new people, and make a lot of new friends, some of whom have become akin to family, and have stood by me through thick and thin.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

We are witnessing a revolution in Big Data. Data is increasingly becoming the most valuable commodity. Big organizations are working with Petabytes of data on a daily basis, and are actively developing infrastructure for handling even more. However, despite our improved capability in generating, storing and processing large amounts of data, much work remains to be done in developing techniques that can help us quickly extract key information from such large collections of data.

As long as we are able to express our information need through some keywords, the search engine technology has matured enough that it can surface the documents of interest from a web-scale corpus in mere seconds. Also, with the close integration of Knowledge-Bases with Question Answering (QA) systems, it is now also possible to get direct answers to a wide range of facts seeking queries like "Who is the president of the United States?", or "Who were the founders of Google?". However, NLP techniques are still to reach a stage in which more abstract searches or questions can be addressed. For instance, currently, no search engine or a QA system can handle queries like "Find documents containing sensitive information.", "Find instances of illegal activity in this collection.", or "Find incriminating information in this collection.". More generally, there are many such scenarios in which people processing the collections are not looking for specific keywords within documents, but are rather looking to organize the documents in some predefined categories, which they can then use for a selective review of the documents.

Technically, it is possible to build document classifiers for these abstract categories, however, therein lies the main problem. The state-of-the-art in document classification are supervised techniques which require the end-user to provide labeled documents for every category of interest. However, such a requirement is very limiting due to the following reasons:

1. The meaning of categories might vary from document collection to document collection. For instance, "sensitive" might have different interpretation in a political corpus vs a medical corpus. Thus, each document collection might require new labeling.

2. Some organizations might not have the resources or the technical know-how to annotate documents and build classifiers.

3. It might not be possible for an organization to share their labeled documents with some third-party to use their document classification technology.

Thus, to truly democratize A.I., and to empower every organization and individual to carry out this basic task, there is a need to reduce our dependence from supervised classification and develop

an effective unsupervised text classification method.

This was precisely the motivation behind the Dataless Classification framework introduced by Chang *et al*. [1], and then extended by Song *et al*. [2]. Chang *et al*. argued that just the natural language description of the target labels is sufficient to classify documents into a predefined label taxonomy with a reasonable accuracy. And thus, theoretically, an individual will only have to provide natural language description of his/her categories of interest, and adapt them with a change of the collection or the domain.

However, Chang *et al*.'s Dataless Classifier system still has a lot of room for improvement. The underlying word embedding method – Explicit Semantic Analysis (ESA) – can be improved in many ways. Our work systematically identifies the limitations of building document representations using ESA, and designs techniques to address those limitations one-by-one. In the process we end up introducing 4 new embeddings: EntityESA, Entity2Vec, Topic2Vec and Word2Concept. The hypothesis here was that if we are able to address those limitations, then we'll be able to bridge the gap between Dataless' performance and the supervised baselines. However, despite our best efforts, we found the original ESA representation to be superior to all the techniques introduced in this work. For some of the methods we have a clear explanation for this observation, however we largely attribute these observations to the lack of proper dataset for evaluating Dataless Classification. We then propose a way to create a new dataset that can be used for future dataless evaluations. However, the new embeddings introduced in this work are general enough, and thus can be of independent interest outside the realm of Dataset Classification as well.

We start off in Chapter 2 with a quick review of Dataless Classification and other related concepts that we'll be needing for our work. We also identify the limitations of ESA in this chapter that lay the foundations for the rest of the chapters. We then outline our techniques and experiments around tackling each of those limitations in the next 3 chapters. We outline our work on using entity embeddings in Chapter 3, densifying ESA embeddings in Chapter 4, and learning topic embeddings in Chapter 5. We then outline some issues with using standard document classification datasets for dataless evaluation, and propose a way to create a new dataset that can be used for future dataless evaluations in Chapter 6. We also identify the future work that can be done to address the additional limitations of ESA in Chapter 6. We then finally conclude in Chapter 7.

CHAPTER 2: BACKGROUND

In this chapter we review the most relevant literature for our work. We start by reviewing the widely popular Word2Vec embedding in section 2.1 which acts as the basic building block for three of the new embeddings that are being introduced in this work. We then review the ESA embedding in section 2.2 which is at the heart of Dataless Classification, and is also the building block for one of the new embeddings being introduced in this work. We then review the Topic Models literature in section 2.3, and outline why Topic Models are not ideal for the scenario being studied in this work. We then review Dataless Classification in section 2.4, which constitutes the main framework that we use throughout this work. We then finally outline the limitations of ESA in section 2.5, which will form the basis for the rest of the chapters.

2.1 WORD2VEC

Word2Vec [3, 4] is a popular method for embedding words into a dense semantic space. At the heart of Word2Vec is the distributional hypothesis i.e. "words that are used and occur in the same contexts tend to purport similar meanings". Word2Vec cleverly applies this hypothesis and creates a semantic space in which contextually similar words are mapped close to each other, and the dissimilar ones are separated from each other. To be able to create such a space, Word2Vec needs a large text corpus (such as Wikipedia) for training.

The training objective of Word2Vec (more specifically, the skip-gram model) is to find word representations that are useful to predict context words given the target word. Formally, given a sequence of K words $w_1, w_2, ..., w_K$, the model aims to maximize the following objective function:

$$L_w = \sum_{k=1}^{K} \sum_{-c \leq j \leq c,\ j \neq 0} \log P(w_{k+j}|w_k) \tag{2.1}$$

where c is the size of the context window, w_k denotes the target word, and w_{k+j} is its context word. The conditional probability $P(w_{k+j}|w_k)$ is computed using the following softmax function:

$$P(w_{k+j}|w_k) = \frac{exp(V_{w_k}^T U_{w_{k+j}})}{\sum_{w \in W} exp(V_{w_k}^T U_w)} \tag{2.2}$$

where W is a set containing all words in the vocabulary, and $V_w \in R^d$ and $U_w \in R^d$ denote the vectors of word w in matrices V and U, respectively. The skip-gram model is trained to maximize the above function L_w. The resulting vectors V are known as the input word embeddings, and U as the output word embeddings. Either of V or U, or some simple arithmetic of them can be used

as the word representation. Figure 2.1 shows the skip-gram architecture.

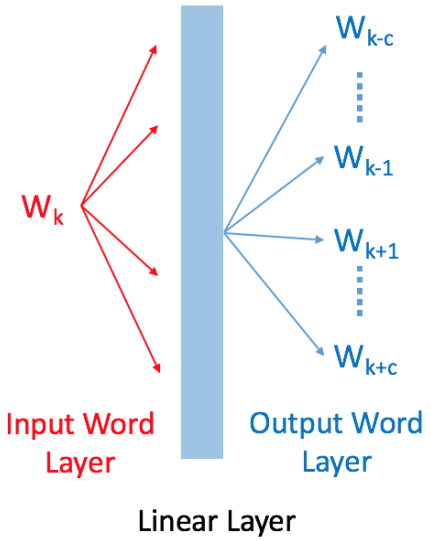

Figure 2.1: Word2Vec skip-gram architecture

2.2 ESA

Explicit Semantic Analysis (or ESA for short) [5] is another popular method for embedding words into a semantic space. ESA defines its semantic space as the space of all Wikipedia articles, and directly embeds words into this semantic space. The main assumption involved here is that each article in Wikipedia corresponds to a concept. It then essentially uses an inverted index for each word to search the Wikipedia corpus, and weighs the retrieved concepts/articles by the TF-IDF scores of the word to form the word representation.

However, ESA is fundamentally different from Word2Vec in the following key aspects:

1. ESA is a sparse representation, whereas Word2Vec is a dense representation. The semantic space of ESA is the space of all Wikipedia articles, and thus for most words, majority of the dimensions will have 0 weight.

2. Unlike Word2Vec where individual dimensions don't have an obvious semantic meaning, ESA's dimensions directly map to a concept in Wikipedia and are thus interpretable.

3. Word2Vec creates a space in which words that occur in similar local contexts get mapped close to each other, whereas ESA maps words close to each other if they are topically similar or occur in similar documents.

2.3 TOPIC MODELING

Topic models [6, 7] offer a formalism for exposing a collection's themes, which can either be used directly by humans to explore the documents of interest, or can be exploited by pipelined applications. Formally, given a document collection, Topic Modeling is the task of discovering topic distributions for each document.

However, traditional Topic models have the following limitations:

1. Topic Modeling approaches usually rely on the presence of a large document collection.
2. Topic models churn out topics which are not easily interpretable for the end-users.
3. Topic Models are widely known to be insensitive to very fine-grained topics, and thus might not be ideal for very specific topic needs.
4. Users often have a set of topics that they care about, however, the Topic Modeling formalism doesn't offer a very natural way of injecting this knowledge into the framework, and thus the end-user is often left with the task of mapping the output topics to their specific interest set.

Recently, there has been some work on incorporating knowledge into topic models [8, 9, 10, 11, 12, 13], however, none of these works is capable of limiting the set of output topics to a controlled set. Intuitively also, it seems rather limiting that Topic Models are completely reliant on the given collection to know everything there is to know about a particular topic. For instance, the given collection should be self-sufficient in itself to be able to identify a very fine-grained topic like **water-shortage**. These key limitations make Topic Modeling unsuitable for the problems where end-users have specific document classification needs, which is precisely the setting that we have set out to explore in this work.

This motivated us to look beyond Topic Modeling for a Knowledge-Driven approach, instead of a Collection-Driven one. More specifically, we would like to be able to classify individual documents into abstract topics of interest without the requirement of that document coming from a large collection. Note that this approach is not at all limiting, since if there is enough signal in the given collection, we can always bootstrap and capture that from the collection. So, the effective problem that we are interested in is "Given a taxonomy of labels, use any resource at disposal to classify a given document into them". Dataless Classification as we discuss next is designed precisely for this scenario.

2.4 DATALESS CLASSIFICATION

At a high-level, Dataless Classification [1, 2] is a simple classification framework that embeds the documents and the target labels in a shared semantic space, and then uses a similarity function (for instance cosine similarity) to classify the documents into the target labels. This it does by finding the label representation that is closest to the target document representation. However, the following 2 key things differentiate Dataless Classification from other techniques:

1. Dataless only requires the end-user to provide natural language descriptions of the labels, which it uses to compute the label representations. This is in stark contrast to supervised learning techniques that require labeled documents for every category of interest.

2. Dataless uses the ESA representation for embedding documents and labels.

Dataless uses the ESA embeddings of all the words in the label descriptions and documents, and composes them (weighted-average) using their TF-IDF scores as weights to compute the label and document representations respectively. Thus, given a target document collection, all one needs is the natural language description of the labels, and Dataless Classification can use that to carry out completely Unsupervised Document Classification.

It is crucial to note that Dataless is a very generic framework, and thus as long as one is able to embed the labels and documents in the same space, it can be used for classification. Thus, it is easy to replace ESA with any other word embedding, for instance Word2Vec, and still use Dataless for Classification. However, in Song *et al*. [2], ESA was found to be superior to other word representations. In addition, if the target labels are arranged in a hierarchy, then the Dataless Classification can enrich the parent label descriptions using the child label descriptions, and can carry out either a Top-Down or Bottom-Up inference for classifying document into labels. Song *et al*. had found Bottom-Up inference to be consistently superior to the Top-Down inference.

2.5 LIMITATIONS OF ESA-BASED DOCUMENT REPRESENTATIONS

Despite the superiority of ESA-Based label and document representations in Dataless benchmarks, it has some key limitations:

1. **Ambiguity:** ESA embeds the words in the documents, however, words can be ambiguous. For instance ‘Michael’ could refer to ‘Michael Jordan’ the Machine Learning Professor or ‘Michael Phelps’ the swimmer. Since ESA has no way of disambiguating this mention of

Michael, its representation for Michael will have flavor of both Swimming and Machine Learning, thus polluting its representation.

2. **Expressivity:** ESA also looks at all the words in the target document to form a document representation, thus diluting the effect of an individual word. However, humans generally don't require all the words of the document to get a sense of the text. For instance, seeing a mention of "Barack Obama" can signal that the document has something to do with Politics. Thus, it might make sense to identify named entities and give them more importance.

3. **Computationally Expensive:** ESA is a sparse representation over a large number of Wikipedia articles ($\approx$ 5M articles), and thus storing all the embeddings in main memory, and computing similarities over this large space is computationally expensive. Thus, the question arises if it is possible to retain the benefits of ESA in a much smaller dense representation.

4. **Heuristic:** ESA is not a learned representation, and thus it relies on the heuristic of using TF-IDF as the dimension weights. Thus, the question arises about the usefulness of TF-IDF, and if it is possible to improve the ESA representation by somehow learning the dimension weights.

5. **Sparsity:** ESA uses the sparse space of all Wikipedia articles, whereas there are ample similarities between individual articles. Thus, intuitively, the signal in ESA is distributed across similar articles. So, the question arises if we can somehow compress these similar articles in a compact space to boost up the signal.

6. **Compositionality:** Averaging ESA word representations to obtain document representations is not ideal since languages exhibit non-compositionality, where for instance, a phrase's meaning is not always derivable from its constituent words. More formally, while composing representations for higher semantic units like phrases, sentences, paragraphs and documents, ESA makes no attempt at understanding the semantic unit and uses trivial composition functions like addition or averaging.

Motivated by this, we hypothesise that it should be possible to improve the performance of Dataless Classification if we can overcome some of these limitations. We thus develop techniques for addressing the *Ambiguity* and *Expressivity* problems in Chapter 3, *Computationally Expensive* related issues in Chapter 4, *Heuristic* and *Sparsity* issues in Chapter 5, and finally propose a way to address the *Compositionality* related issues in Chapter 6, which can be pursued in the future.

CHAPTER 3: REFINING EMBEDDINGS USING ENTITIES

In this chapter, we set out to address the *"Ambiguity"* and *"Expressivity"* related limitations of ESA. The key limitation here stems from the fact that ESA is essentially a representation for words, and words can be ambiguous.

One possible way to address this limitation is to move towards phrases or n-grams, however they too can suffer from the problem of ambiguity. For instance, a phrase "Michael Jordan" can potentially refer to Michael Jordan, the Basketball Player or to Michael I. Jordan, the Berkeley Professor. Clearly, such diverse possible interpretations of the phrases/mentions will pollute their representations, which will then propagate to representations of higher semantic units that utilize such embeddings. So, there is a need of disambiguating such ambiguous mentions in text before their representations are constructed and subsequently used in a target text/application. The techniques for disambiguating ambiguous entity mentions are widely studied as Entity Linking. Once such mentions are accurately disambiguated, new cleaner embeddings can be constructed for the disambiguated mentions by using Word2Vec or ESA like formulations.

We start with a brief overview of Entity Linking in section 3.1, and then in section 3.2 introduce a new embedding that uses an ESA like formulation for creating entity embeddings, we call it **EntityESA**. Similarly, we introduce **Entity2Vec** in section 3.3 as another new entity embedding that instead uses a Word2Vec like formulation. We then explain our experimental setup and report results in section 3.4

3.1 ENTITY LINKING

Entity Linking [14, 15, 16, 17, 18, 19, 20] is the task of linking (or disambiguating) an ambiguous mention of a named entity in a document to its corresponding entry (entity) in a Knowledge-Base such as Wikipedia. Since multiple entities in the KB can be referred to by the same mention, the context of the mention is used in discerning the intended entity (or sense) from the rest. For instance, the mention *Michael Jordan* can be used for multiple people in the KB, but, if the mention occurs in the vicinity of words like "Championship", "NBA", or "player", it probably refers to Michael Jordan – the basketball player. However, if the mention has words like "Machine Learning", "Andrew Ng", or "Berkeley" in its vicinity, it most likely is intended for Prof. Michael I. Jordan – the Berkeley Professor. Entity Linking has a very rich literature with techniques consisting of various forms of local and global models. One of the most popular systems is the "Illinois Wikifier" from Ratinov *et al*. [14], which also has a public implementation and demo. In this work, we rely heavily on this system for disambiguating mentions, and creating our entity embeddings.

3.2 ENTITY ESA

We propose **EntityESA** as a very natural extension of ESA for entities. We'll refer to the word-level ESA as **WordESA** henceforth. We first use an Entity Linking system to disambiguate ambiguous mentions in Wikipedia. Once we have such disambiguated entities in a Wikipedia document, we adapt the WordESA embedding construction procedure for entities. That is, instead of creating a TF-IDF weighted inverted index of words, we create a TF-IDF weighted inverted index of disambiguated entities. Now, instead of looking up a word in this inverted index, we'll be looking up an entity, and will use the TF-IDF of that entity in Wikipedia as the dimension weight. Furthermore, since most of the Entity Linking systems output a confidence score for their output disambiguations, we can create different EntityESA embeddings using different disambiguation confidence thresholds.

The promise of this embedding comes from two different aspects:

1. **Reduction of noise:** Firstly, disambiguation of mentions will ensure that multiple interpretations of a word won't contribute to a single word representation, but in fact will contribute only to the representation of the intended meaning/entity. For instance, the occurrences of the swimmer "Michael", and basketball player "Michael" will not pollute the representation of the berkeley professor "Michael Jordan".

2. **Context Enrichment:** Secondly, disambiguation of mentions will also ensure that different ways of mentioning a particular entity will start contributing to a single representation instead of distributing it across various surface-form representations. For instance, all different ways of mentioning "Barack Obama", be it "Barack", "Obama", or the "44th President of U.S.A" will contribute to the representation of the entity "Barack Obama".

Both of these factors will ensure that we get cleaner and richer representations as compared to WordESA.

3.2.1 Using EntityESA with Dataless

The entity embeddings obtained using the methodology above, along with the word-level ESA embeddings (WordESA), act as the basic building blocks for the new representations that can be used with Dataless Classification. We have various options for constructing the document and label representations now:

Document Representations: To utilize the entity embeddings for constructing clean document embeddings, we need to first run the entity linking system on the target document to disambiguate the ambiguous mentions. The document will then consist of words and disambiguated entities. We now have 2 options for constructing the document representations:

1. **EntityESA-Only:** In this approach, we just use the EntityESA representations for the disambiguated entities in the target document, and combine them with a suitable composition function (such as TF-IDF weighted average) to form the document representations. Note that, in this approach, only the entities contribute to the document representation, and words are completely ignored. This approach for constructing the document representation might not be ideal for those scenarios where the entity linking system fails to identify many entities within the document. Thus, to handle such scenarios, we also propose **ComposedESA**.

2. **ComposedESA:** Both EntityESA and WordESA embed entities and words respectively in the space of all wikipedia articles. This allows us to combine the entity and word representations using some composition function (such as weighted averaging) to form the final document representation. For instance, while creating the document representation, we might want to give more importance to the entity representations. This can be done by giving a higher weight to the entity embeddings while computing a weighted average of the entity and word embeddings,

Label Representations: Similar to the document representations, we have the following options for constructing the label representations:

1. **WordESA-Only:** Using only the word embeddings, similar to the original Dataless Classification work by Chang *et al*. [1]

2. **EntityESA-Only:** Using only the entity embeddings, similar to the document representation section above.

3. **ComposedESA:** Using a combination of word and entity embeddings, similar to the document representation section above.

However, note that in order to be able to use the entity embeddings for constructing the label representations, the end-user will have to do some extra work:

- They'll either have to manually (or fuzzily) map their labels to wikipedia entities, or

- They'll have to provide some natural language description about the label (and not just the keywords), which we can then treat as a document and run our entity linking system on.

Note that the additional cost of manually linking a label to Wikipedia can be easily overcome by using that label's Wikipedia page to embellish the label description/representation, which as we report in section 3.4 can improve the results many-folds.

Once we have the document and label representations from above, we can now use the Dataless framework for classification using a similarity metric such as cosine. Note that this framework is exactly similar to the original work from Chang *et al.* [1], and thus Co-Training, Bootstrapping and Hierarchical Classification can be used here as well to augment the simple classification.

3.3 ENTITY2VEC

We propose **Entity2Vec** as a very natural extension of Word2Vec for entities. Similar to EntityESA, we first use an entity linking system to disambiguate ambiguous mentions in Wikipedia. We can now adapt the word2vec embedding construction procedure for entities. We do this by simply considering the disambiguated entities as new tokens for our vocabulary, and running the word2vec training procedure in its original form. Thus, entity2vec is essentially word2vec with the basic modification that the tokens in training documents now consist of words as well as disambiguated entities.

The training objective of the Entity2Vec model is to find word and entity representations that are useful to predict context words and entities given the target word or entity. Formally, given a sequence of K "units" $z_1, z_2, ..., z_K$, where we define a unit z_k as the word w_k if the kth sequence element is a word, or as the entity e_k if the kth sequence element is an entity, the model aims to maximize the following objective function:

$$L_z = \sum_{k=1}^{K} \sum_{-c \leq j \leq c,\ j \neq 0} \log P(z_{k+j}|z_k) \tag{3.1}$$

where c is the size of the context window, z_k denotes the target unit, and z_{k+j} is its context unit. Here, the unit z_i can be either a word w_i or an entity e_i. The conditional probability $P(z_{k+j}|z_k)$ is computed using the following softmax function:

$$P(z_{k+j}|z_k) = \frac{exp(V_{z_k}^T U_{z_{k+j}})}{\sum_{z \in Z} exp(V_{z_k}^T U_z)} \tag{3.2}$$

where Z is a set containing all units in the vocabulary, where $Z = W \cup E$, and W and E are the word and entity vocabularies respectively. $V_z \in R^d$ and $U_z \in R^d$ denote the vectors of the unit z in matrices V and U, respectively. The entity2vec model is trained to maximize the above function

L_z. The resulting vectors V are called the input unit embeddings, where V has both word and entity embedding components. Similarly, the vectors U are called the output unit embeddings, where U has both word and entity components as well. Figure 3.1 shows the Entity2Vec architecture.

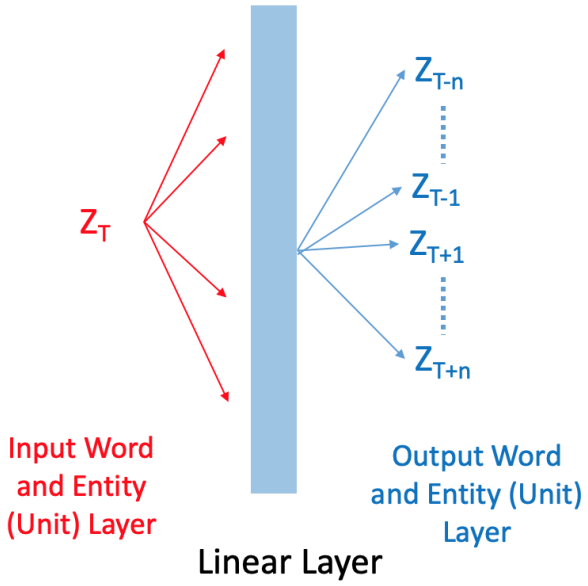

Figure 3.1: Entity2Vec architecture

Similar to EntityESA, Entity2Vec has the advantages of "Reduction in noise" and "Context Enrichment" over vanilla Word2Vec. Through this objective, we hope to be able to improve both the word and entity representations. The word representations will improve because of the word-entity interactions, specifically, because of the presence of unambiguous disambiguated entities in the context of the words. Whereas, the entity embeddings will improve directly because of the entity-entity interactions, wherein disambiguated entities in the vicinity of each other improve each other's representations. The entity embeddings will also improve because of the combination of entity-word and word-word interactions. The word-word interactions will help map synonyms words closer to each other, which the entity representations will then benefit from through the entity-word interactions.

3.3.1 Document Classification using Entity2Vec

At this point, it is crucial to point out some additional key differences between the ESA-Based and Word2Vec-Based representations:

- Word2Vec and Entity2Vec are dense representations that embed words and entities respectively in a dense non-interpretable space. Furthermore, since the space is non-interpretable,

and different dense representations were not trained to embed words and entities in the same space, they can't be easily combined with each other.

- In contrast, both WordESA and EntityESA are sparse representations that embed the words and entities respectively in the interpretable space of wikipedia articles. Since the space is shared between different representations, and is interpretable, they can be easily combined with each other.

Since Entity2Vec is a dense representation, it can't be easily combined with other dense or sparse representations. This results into the following limited options for computing document and label representations:

Document Representations: Similar to EntityESA, to be able to use the new entity embeddings, we need to first run an entity linking system on the target document to disambiguate the entities. However, since Entity2Vec can't be combined trivially with another word embedding, we can only combine the entity embeddings with each other using some composition function (such as weighted average) to construct the document representation. Note that, the words are completely ignored here, which as was pointed out earlier, can be very limiting in cases where the entity linking system identifies very few entities in the target document.

Label Representations: Similar to EntityESA, the end-user will have to do the additional task of either manually (or fuzzily) linking the labels to Wikipedia, or providing a natural language description of the label that can be fed to the entity linking system for disambiguating the entities. Furthermore, similar to the limited options for creating the document representations, we can only use the entity embeddings with some simple composition function (such as weighted average) to create the label representations.

3.4 EXPERIMENTS & RESULTS

In the absence of a well-defined task to carry out an intrinsic evaluation for our new entity embeddings, we used Dataless Classification for an extrinsic evaluation. For the Dataless experiments, we used the 20Newsgroups dataset, which contains over 20K documents, with 26 labels. We used the 2-Level label hierarchy from Song *et al*. [2] in which 6 labels are at the top of the hierarchy, whereas the rest 20 are organized at Level-2. We also used the embellished label descriptions from Song *et al*., and used the bottom-up tree traversal for inference. We used averaging as the composition function over the embeddings. We computed Micro-F1 and Macro-F1 using all the 26 labels in the hierarchy. Also, unless explicitly stated otherwise, we used 500 dimensions

for all ESA experiments.

For disambiguating the mentions, we used the Illinois-Wikifier system from Cheng *et al.* [20] We used it for annotating the following two corpora:

- The 20newsgroups dataset, so that we can evaluate the new entity embeddings.
- A 2013 version of Wikipedia which contained around 4.5M articles. This annotated Wikipedia was used for creating our Entity2Vec and EntityESA embeddings.

Annotating the entire Wikipedia using Illinois-Wikifier took around 3-4 weeks on a single server. We created 6 different EntityESA embeddings, one each for 0.0, 0.1, 0.3, 0.5, 0.7 and 0.9 as disambiguation confidence thresholds, and stored them using Lucene. To benchmark our embeddings, we re-implemented the Dataless Classification system from Song *et al.*, and integrated it with the Cogcomp-NLP library [21]. We have released our implementation for public consumption[1]. Furthermore, we extended the said implementation for experimenting with our new entity embeddings, and have released the corresponding code as well.[2]

For training Entity2Vec, we used the original Word2Vec implementation[3] and the default hyperparameters from Mikolov *et al.*

3.4.1 Entity2Vec vs EntityESA

Since training different version of Entity2Vec on the entire Wikipedia using different confidence thresholds was too expensive, we decided to first compare the two entity representations built using the same confidence threshold with each other and select a winner between the two. For this purpose, we used a confidence threshold of 0.0, and compared the performance of EntityESA and Entity2Vec. Since Entity2Vec can only work with a label description created using the entities, we manually mapped the label descriptions (keywords) to entities in Wikipedia, and used the corresponding entity representations for computing the label representations. We report the Micro-F1 and Macro-F1 numbers for Dataless Classification on the 20newsgroups dataset in Table 3.1. We also provide the Word2Vec numbers for reference. We used 300 dimensions for all the embeddings. An embedding size of 300 for an ESA-like representation means that we'll only keep around the top 300 dimensions for each word. Note that, the top 300 dimensions might vary

[1] https://github.com/CogComp/cogcomp-nlp/tree/master/dataless-classifier
[2] https://gitlab-beta.engr.illinois.edu/sgupta96/datalessclassification
[3] https://github.com/tmikolov/word2vec

from word to word.

Method	Word2Vec	Entity2Vec	EntityESA
Micro-F1	0.42	0.41	0.51
Macro-F1	0.30	0.40	0.48

Table 3.1: Performance of different Entity Embeddings (300 dim.)

The table shows that both entity representations are better than Word2Vec, thus showing the power of clean entity representations. Between Entity2Vec and EntityESA, EntityESA is a clear winner. This again shows the superiority of ESA-Based sparse representations over Word2Vec-Based dense representations in Dataless Classification. Since EntityESA was a clear winner here, we discarded Entity2Vec and carried more detailed experiments with only EntityESA.

3.4.2 EntityESA-Only as Document Representation

Our first set of experiments with EntityESA involved trying to quantify the impact of different disambiguation confidence thresholds. For this purpose, we used only the entities in the documents for computing the document representations, and used WordESA with the same set of keyword descriptions for labels that was used by Song *et al*. for computing the label descriptions. We report the corresponding numbers in Table 3.2.

Threshold	0.1	0.3	0.5	0.7	0.9
Micro-F1	0.49	0.52	0.55	0.56	0.57
Macro-F1	0.43	0.45	0.47	0.48	0.48

Table 3.2: Impact of thresholds on EntityESA

As can be seen from the table, EntityESA representations constructed using higher disambiguation confidence thresholds outperform the ones created using the lower confidence thresholds. This is very easy to explain since with the increment in the confidence threshold, the entity embeddings are expected to be cleaner and sparser.

We then benchmarked our best performing EntityESA configuration (0.9 threshold) against vanilla WordESA. We report the corresponding numbers in Table 3.3.

Method	WordESA	EntityESA
Micro-F1	0.65	0.57
Macro-F1	0.56	0.48

Table 3.3: Performance of EntityESA 0.9

As can be seen from the table, EntityESA performs worse than WordESA. This was a surprising outcome for us. Our initial hypothesis was that this is probably because of the fact that we only use the entity embeddings and completely disregard the words for computing the document representations. Thus, the document representations can suffer where the entity linking system fails to identify many entities. This led us to carry out experiments with our earlier proposed solution of using ComposedESA instead, i.e. using both word and entity representations with a suitable composition function.

3.4.3 ComposedESA as Document Representation

We first wanted to verify that the trend that increasing the confidence threshold improves the performance holds with ComposedESA as well. For this purpose, we used ComposedESA for computing the document representations. We used equal weights for both the WordESA and EntityESA components. And like before, we used the WordESA embeddings for computing the label representations. We report the corresponding numbers in Table 3.4.

Threshold	0.1	0.3	0.5	0.7	0.9
Micro-F1	0.52	0.55	0.57	0.58	0.58
Macro-F1	0.45	0.48	0.49	0.50	0.50

Table 3.4: Impact of thresholds on ComposedESA

As can be seen in the table, the trend with the confidence thresholds holds for ComposedESA as well.

We then benchmarked the best performing ComposedESA configuration (0.9 threshold) against WordESA. We report the corresponding numbers in Table 3.5.

As can be seen in the table, ComposedESA only marginally increases the performance over EntityESA, and still performs much inferior to WordESA. This clearly pointed in the direction that the EntityESA component itself is the problem.

Method	WordESA	EntityESA	ComposedESA
Micro-F1	0.65	0.57	0.58
Macro-F1	0.56	0.48	0.50

Table 3.5: Performance of ComposedESA 0.9

We also experimented with the Top-Down hierarchical classification approach, as compared to a Bottom-Up approach that we have been using until now in all of our experiments. We report the corresponding numbers for EntityESA and ComposedESA in Tables 3.6 and 3.7 respectively.

Method	Bottom-Up	Top-Down
Micro-F1	0.57	0.51
Macro-F1	0.48	0.45

Table 3.6: Bottom-Up vs Top-Down EntityESA

Method	Bottom-Up	Top-Down
Micro-F1	0.58	0.53
Macro-F1	0.50	0.46

Table 3.7: Bottom-Up vs Top-Down ComposedESA

As can be seen in the tables, Bottom-Up inference outperformed the Top-Down inference. This is similar to the trend that was observed in Song *et al*. This validated our hypothesis that the Tree-Traversal method was not the reason for a drop in performance.

3.4.4 EntityESA as Label Representation

Note that, all the experiments above were only changing the document representations, whereas the label representations were fixed to be only the WordESA ones. Thus, before digging deeper into the EntityESA component, we also wanted to evaluate the impact of using entity embeddings in the label description as well. Towards this goal, we manually mapped the keyword description of labels to multiple entities in Wikipedia, and used the EntityESA embeddings for those entities to be the label representations. We report the corresponding numbers in Table 3.8.

As can be seen in the table, the performance further degraded. Thinking that sparsity of the label representation is hurting us here, we experimented with embellishing the label representations further. Towards this goal, we ran an entity linking system on the first paragraph of the linked entity

Label Rep.	WordESA	EntityESA
Micro-F1	0.57	0.36

Table 3.8: Performance of EntityESA with different Label Rep.

page to embellish the label representation and noted significant gains. We report the corresponding numbers in Table 3.9. This observation is similar to Song *et al.*, where label embellishment was found to be useful too. Unfortunately, the performance boost reached the ceiling after a while, and even then it couldn't match the performance of using WordESA as the label representation.

Label Rep.	WordESA	EntityESA Original	EntityESA Embellished
Micro-F1	0.57	0.36	0.50

Table 3.9: Impact of Label Embellishment with EntityESA

3.4.5 Diagnosis

Detailed experiments carried out in the previous section convinced us that even after the theoretical advantages, practically, the EntityESA embedding is not able to beat the performance numbers of the WordESA embedding. This prompted us to look at an incorrectly classified document to diagnose the problem. We identified a document which originally belonged to the label **comp.windows.x**, but was being classified into **soc.religion.christian**. We noted (Table 3.10) the document representation from EntityESA to be significantly inferior to that from WordESA. For a document labeled with a Computers category, the WordESA representation seems to be capturing computers related concepts in the Top-5 dimensions, whereas the Top-5 concepts from EntityESA seem to be way-off.

WordESA	EntityESA
Comparison of C Sharp and Java	Educational technology
Standard Widget Toolkit	Mental disorder
X Window System	Deadwood characters
Operating system	Anabolic steroid
Abstraction	Breastfeeding
Comparison of Java and C++	Contract

Table 3.10: Document Rep. for a **windows.x** document

We then further looked at the output of the entity linking system for this document, and noted a lot of erroneous disambiguations. We observed that some of the email addresses were leading to

errors. For instance, **michael@Imp.HellNet.org** and **devil@loki.HellNet.org** were being linked to Hell, Imp, Devil and Loki, which were driving the entire document representation towards that of Religion. However, we found that this was not the only pattern that was causing problems, and that there were several other unforced disambiguation errors. This clearly pointed towards the root of the problem, that our classification task was suffering due to the entity linking errors. We had anticipated such errors, but were of the opinion that probably a limited number of errors will get balanced by the other correct annotations, however, since most of the state-of-the-art entity linking systems exploit Global Coherence within the document, it makes sense that a couple of wrong annotations are impacting the entire document.

3.4.6 Qualitative Analysis of EntityESA Embedding

We noted in the previous section that using EntityESA for Dataless suffers due to the inaccuracies of the entity linking system. This however, doesn't say anything about the quality of the EntityESA embeddings themselves. It is understandable that an entity linking system will make mistakes on a news corpus such as 20newsgroups since it is trained on Wikipedia and possibly suffers due to domain transfer problem. However, entity linking systems are known to perform well on Wikipedia, and thus our EntityESA embedding might still be of high quality.

We manually inspected a couple of EntityESA embeddings and found them to be extremely superior to the WordESA embeddings. We provide an example in Table 3.11 for the earlier mentioned example of "Michael I. Jordan" – the Berkeley Professor. The top 5 concepts from WordESA correspond to different semantic interpretations of Jordan. However, the EntityESA embedding retrieves only those concepts that are directly related to the Berkeley Professor, in this case some of the work by the Professor.

WordESA	EntityESA
Air Jordan	Computational creativity
Michael Jordan	Hierar. Dirichlet process
Jordan River	Recurrent neural network
Hal Jordan	Topic model
Jordan (Country)	Internal model

Table 3.11: Entity Rep. for **Micheal I. Jordan**

In addition to this basic qualitative analysis, we also computed relatedness scores between various entities by using Jaccard as the similarity metric between their representations (max dimension size 100). We found the similarity score between an entity pair to be high when the entities are related to each other, and it to be low when they are not related to each other. This also supports

our hypothesis that EntityESA representations are clean and sparse. We provide some examples in Table 3.12.

Entity 1	Entity 2	Jaccard
Michael Jordan	Chicago Bulls	0.25
Michael I. Jordan	Chicago Bulls	0.0
Steve Jobs	Apple Inc.	0.20
Steve Jobs	Apple (fruit)	0.0
Larry Page	Sergey Brin	0.42

Table 3.12: Entity Relatedness using EntityESA

As can be seen in the table, the Jaccard similarity between “Michael Jordan” the basketball player and the “Chicago Bulls” team is higher as compared to the similarity between ”Michael I. Jordan” the Berkeley Professor and “Chicago Bulls”. Similarly, “Steve Jobs” and ”Apple” the company have more similarity as compared to “Steve Jobs” and the ”Apple (fruit)”.

CHAPTER 4: DENSIFYING ESA

In this chapter, we set out to address the *"Computationally Expensive"* related limitations of ESA. These issues essentially stem from the fact that ESA is a sparse representation over a space of 5M dimensions, where for most words, a large number of dimensions have 0 weight. Not only does sparse representations consume more main memory, but it is also computationally expensive to do any arithmetic with them. However, ESA-Based sparse representations have been shown to be superior to Word2Vec-Based dense representations, thus it is worthwhile to investigate if it is possible to retain the benefits of ESA in a dense representation, which we refer to as **Densification** henceforth.

Intuitively, it should be possible to do so, since there's a lot of redundancy in Wikipedia articles. We probably don't need all 5M dimensions to represent most words, and thus it should be possible to compress a lot of dimensions. In this chapter we discuss some heuristics that we tried towards this goal. Then, in Chapter 5 we outline our attempts at learning to compress the dimensions.

We start in section 4.1 with a description of our random projection ideas. Then, in section 4.2 we outline our ideas around using other representations for densifying ESA. We finally describe our experimental setup and share results in section 4.3.

4.1 USING RANDOM PROJECTION

Let $E_{W \times D}$ be a matrix consisting of ESA embeddings of W words of dimensionality D each, where e_{ij} is the j^{th} dimension of the i^{th} word w_i. Also, let $R_{D \times K}$ be a matrix where each cell value is drawn randomly from some distribution. We now define $M_{W \times K}$ as the matrix multiplication of the ESA embedding matrix E and the random matrix R from above:

$$M = E \times R \tag{4.1}$$

When $K \ll D$, M can be considered as a densified version of the original ESA embedding matrix E. Intuitively, we are defining new dimensions by randomly sampling the compositions of the existing ESA dimensions. More specifically, the j^{th} column of R contains the composition code that will be applied to the original ESA dimensions to produce the j^{th} dimension of the densified ESA. Thus, intuitively, we are defining new concepts as a composition of the old concepts.

We have different distribution options for sampling cell values in R. We experimented with the

following:

- **Binary Projection:** We randomly sample each cell value from the set $\{-1, 0, 1\}$. This is equivalent to randomly deciding whether a concept should be ignored, subtracted or added to form a new concept.
- **Uniform Projection:** We sample each cell value from a uniform distribution $U(0, 1)$.
- **Gaussian Projection:** We sample each cell value from a Gaussian Distribution $\mathcal{N}(0, 1)$

One of the biggest disadvantages of the random projection idea is that there is no guarantee that the new dimensions will be useful or meaningful at all. While composing the old dimensions randomly, we could be cancelling out the effect of similar dimensions or could be suppressing really useful dimensions. Thus, the idea of random projection can be considered as a baseline for any other method that tries to compress the dimensions in a principled manner.

4.2 USING OTHER REPRESENTATIONS

One of the main advantages of ESA is that its dimensions are interpretable, since they directly map to Wikipedia articles. Thus, it makes sense to try to use this property of ESA, instead of randomly combining/compressing the dimensions. Thus, if we can somehow map the individual dimensions/concepts of ESA to a dense space wherein similar concepts are mapped close to each other, we might be able to compress the ESA space better. As it turns out, dense embeddings are designed specifically for this purpose. Thus, we propose the following method for densifying the ESA representation:

1. **Step 1:** Create the ESA representation for the target word/document. Let's call it the *Parent* embedding.
2. **Step 2:** Look up the dense embeddings for each concept/dimension in the ESA representation from above. Let's call these the *child* embeddings.
3. **Step 3:** Do a weighted-average of the child embeddings obtained above, where the weight is the original dimension value in the parent embedding – i.e. the TF-IDF score of the ESA dimension.

We now have several options for the child embeddings:

- **Dense Word Embedding:** Such as Word2Vec.

- **Dense Entity Embedding:** Such as Entity2Vec, as was introduced in the previous section.
- **Sparse Entity Embedding:** Since this framework is quite expressive, we can even use a sparse entity embedding such as EntityESA. The promise of this method lies in the fact that EntityESA is a cleaner and sparser representation, and thus we'll not only be able to create a richer representation, but will also be able to save some computation time because of the additional sparsity of the representation.

It is important to point out here that this framework is flexible enough that the dimensionality of the parent embeddings and the child embeddings need not match. For instance, we can densify a 1000 dimensional parent ESA embedding using a 500 dimensional child Word2Vec embedding, or even a 100 dimensional parent ESA embedding with a 1000 dimensional child Word2Vec embedding.

4.3 EXPERIMENTS & RESULTS

We used the same experimental setup as in the previous chapter. We evaluate the new dense embeddings on the task of Dataless Classification on 20newsgroups corpus, using bottom-up inference, and the embellished label representations from Song *et al*. We calculate Micro-F1 and Macro-F1 numbers over all 26 labels. Like before, we are also making our implementations public.[1]

4.3.1 Random Projection

We report the numbers for the new dense randomly projects embeddings with dimensionality 500 in Table 4.1, and with dimensionality 1000 in Table 4.2. We also report the corresponding ESA and Word2Vec numbers for reference.

Method	Sparse-ESA	Word2Vec	Binary-ESA	Gaussian-ESA	Uniform-ESA
Micro-F1	0.69	0.52	0.17	0.57	0.57
Macro-F1	0.62	0.45	0.07	0.51	0.52

Table 4.1: Random Projection Performance (500 dim.)

As can be seen in the tables, Binary random projection performs much inferior to the Gaussian and Uniform projections. Gaussian and Uniform projection performances are very close to each

[1] https://gitlab-beta.engr.illinois.edu/sgupta96/datalessclassification

Method	Sparse-ESA	Word2Vec	Binary-ESA	Gaussian-ESA	Uniform-ESA
Micro-F1	0.69	0.54	0.19	0.63	0.65
Macro-F1	0.62	0.48	0.09	0.57	0.58

Table 4.2: Random Projection Performance (1000 dim.)

other, with the uniform projection being marginally better. However, none of the projection ideas is able to beat the Sparse ESA numbers. This was expected since the densification ideas were not aimed at outperforming ESA's F1 numbers, but were instead aimed at improving its computational footprint while trying not to lose much on the performance. The drop in performance is significant with 500 dimensions, however, the gap in performance reduces when the number of dimensions is increased to 1000. Interestingly, both Gaussian and Uniform projection ideas perform better than Word2Vec, which signals that these new dense embeddings are able to retain some of the benefits of the sparse ESA.

4.3.2 Word2Vec

We report the numbers for densification of ESA embeddings of size 500 using Word2Vec embeddings of different sizes in Table 4.3. We also report the corresponding numbers for ESA, Word2Vec and Uniform projection embeddings of size 500 for reference.

#Dimensions	ESA	Word2Vec	Uniform	100	500	1000
Micro-F1	0.69	0.52	0.57	0.50	0.55	0.55
Macro-F1	0.62	0.45	0.52	0.40	0.46	0.46

Table 4.3: Densification with Word2Vec (P: 500 dim, C: Variable)

As can be seen in the table, the performance goes up with an increase in the size of the Word2Vec embedding, however, the gain saturates after a while. Densified representation of comparable size performs better than plain Word2Vec. This shows that the new densified representation is able to retain some benefits of vanilla ESA. However, it performs inferior to ESA, and even to the uniformly projected baseline. This observation can be attributed to the fact that the Word2Vec representation being used for densification was itself not a very useful representation for the Dataless task.

4.3.3 Entity2Vec

We report the numbers for densification of ESA embedding of size 500 using Entity2Vec embedding of size 300 in Table 4.4. We also report the corresponding numbers for ESA, Word2Vec and Uniform projection embeddings of size 500 for reference.

Method	ESA	Word2Vec	Uniform	Entity2Vec
Micro-F1	0.69	0.52	0.57	0.55
Macro-F1	0.62	0.45	0.52	0.46

Table 4.4: Densification with Entity2Vec (P: 500 dim, C: 300 dim)

As can be seen in the table, the densified representation performs better than plain Word2Vec. This shows that the new densified representation is able to retain some benefits of vanilla ESA. However, it performs inferior to ESA, and even to the uniformly projected baseline. This observation can be attributed to the poor performance of Entity2Vec itself.

4.3.4 EntityESA

We first compare the performances of the two Entity embeddings – Entity2Vec and EntityESA. We report the numbers for densification of ESA embedding of size 500 using entity embeddings of size 300 in Table 4.5.

Method	Entity2Vec	EntityESA
Micro-F1	0.55	0.59
Macro-F1	0.46	0.50

Table 4.5: Comparison between Entity Embeddings (P: 500 dim, C: 300 dim)

As can be seen in the table, EntityESA is a clear winner between the two. This was expected since EntityESA on its own was found to be superior to Entity2Vec in the previous chapter.

To benchmark the performance of EntityESA for densification, we also report the numbers for densification of ESA embedding of size 500 using EntityESA embedding of different sizes in Table 4.6. We also report the corresponding numbers for ESA, Word2Vec and Uniform projection embeddings of size 500 for reference.

As can be seen in the table, the performance of EntityESA based densification goes up with an increase in the size of the EntityESA embedding. Also, densified representation of comparable size performs better than plain Word2Vec. This shows that the new densified representation is able to

#Dimensions	ESA	Word2Vec	Uniform	100	500
Micro-F1	0.69	0.52	0.57	0.57	0.59
Macro-F1	0.62	0.45	0.52	0.48	0.50

Table 4.6: Densification with EntityESA (P: 500 dim, C: Variable)

retain some benefits of vanilla ESA. The new representation is also comparable or marginally better than the uniform projection based baseline. However, it performs inferior to ESA. This probably can be attributed to the fact that the ESA representation itself suffers because of the ambiguity of words, and using EntityESA representation of the unrelated concepts probably further amplifies the errors. Another reason could be our usage of averaging as the function for composing the representations for densification.

CHAPTER 5: LEARNING TO EMBED TOPICS

In this chapter, we set out to address the *"Heuristic"* and *"Sparsity"* related limitations of ESA. To be more specific, we aim to achieve the following:

1. Learn the weights of the dimensions instead of using the TF-IDF heuristic.
2. Make the representation compact by merging similar dimensions/concepts with each other.
3. Keep the benefits of ESA.

It is important to point out that achieving 3 above is really crucial, since 1 and 2 can be achieved by any dense embedding. It was reported in Song *et al*. [2] that ESA outperforms all other dense embeddings, most notably the Word2Vec [3] embedding. This can be attributed to the fact that Word2Vec is a kind of local embedding that is good at capturing local context, as compared to ESA which captures the topical/global context.

Thus, to achieve all 3 goals above, we should be able to embed topics/concepts in a dense representation. We do so by modifying the Word2Vec objective to also optimize for Topic prediction. When we only optimize for Topic prediction, we get our new embedding **Topic2Vec**. However, when we optimize for both local context and topics, we get our new embedding **Word2Concept**.

We formally describe Topic2Vec in section 5.1, and Word2Concept in section 5.2. Then in section 5.3, we describe how the dataset was created for training these embeddings. We then quickly outline the related work in section 5.4, and talk about our experimental setup and results in section 5.6.

5.1 TOPIC2VEC

Topic2Vec modifies the word2vec skip-gram architecture to embed the topics. The training objective of the Topic2Vec model is to find word representations that are useful to predict the document topics given the target word in the document. Formally, given a document containing K words $w_1, w_2, ..., w_K$ in sequence, and labeled with S Topics $t_1, t_2, ..., t_S$, the model aims to maximize the following objective function:

$$L_t = \sum_{k=1}^{K} \sum_{s=1}^{S} \log P(t_s|w_k) \tag{5.1}$$

where the conditional probability $P(t_s|w_k)$ is computed using the following softmax function:

$$P(t_s|w_k) = \frac{exp(V_{w_k}^T U_{t_s})}{\sum_{s \in T} exp(V_{w_k}^T U_{t_s})} \tag{5.2}$$

where T is the set of all topics in the system, and $V_w \in R^d$ denotes the vectors of word w in matrix V and $U_t \in R^d$ denote the vectors of topic t in the matrix U. The Topic2Vec model is trained to maximize the above function L_t. Like before, the resulting vectors V will be referred to as the input word representation, and vectors U as the output topic representation. Figure 5.1 shows the Topic2Vec architecture.

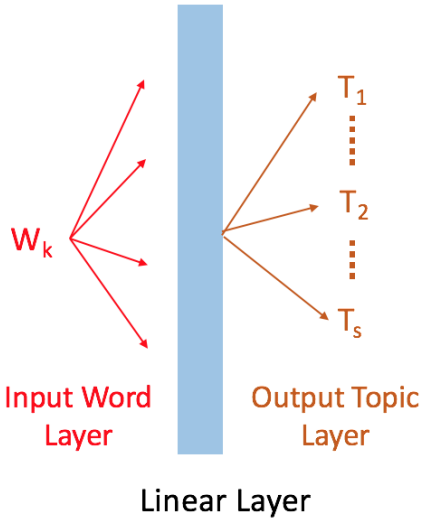

Figure 5.1: Topic2Vec architecture

Intuitively, this objective will push the representation of topics that contain similar words close to each other, and will also push the representation of words that occur in similar documents towards each other.

5.2 WORD2CONCEPT

Word2Vec has been shown to be good at mapping contextually similar words near to each other, whereas Topic2Vec is supposed to map Topically similar words near to each other. Since both of these embeddings capture different yet useful information, it makes sense to try to capture both effects in a single embedding. To achieve this goal, we propose a new embedding called **Word2Concept**. The training objective of the Word2Concept model is to find word representations that are useful to predict both the document topics and the context words, given the target

word in the document. Formally, given a document containing K words $w_1, w_2, ..., w_K$ in sequence, and labeled with S Topics $t_1, t_2, ..., t_S$, the model aims to maximize a weighted sum of the previous two objectives:

$$L_h = L_w + \lambda L_t \tag{5.3}$$

where L_w is the word2vec objective function from equation 2.1, L_t is the topic2vec objective function from equation 5.1, and λ is a hyperparameter that controls the relative importance of the two objectives. The Word2Concept model is trained to optimize the above function L_h. The resulting vectors V will be referred to as the input word embeddings, and vectors U will have two components, one for the target words and another for the target topics; we'll refer to them as output word embeddings and output topic embeddings respectively. Figure 5.2 shows the Word2Concept architecture.

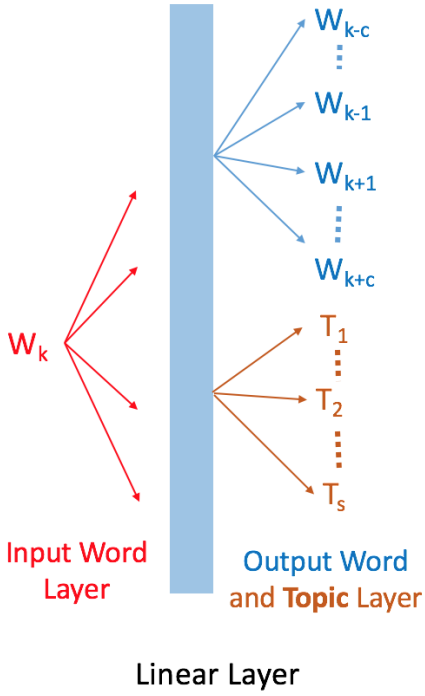

Figure 5.2: Word2Concept architecture

Through this objective, we hope to be able to bring the best of both worlds together, by pushing the embeddings of synonyms words towards each other through the word2vec component, and the embeddings of topically similar words towards each other through the topic2vec component.

5.3 TRAINING DATASET CREATION

In stark contrast to word2vec, which only requires a collection of documents, both of our new proposed embeddings additionally require documents to be labeled with topics. To support this additional information need for our embeddings, we propose a distant supervision based approach.

Wikipedia is the biggest encyclopedia in the world, which contains millions of articles. In addition to the article pages, Wikipedia also contains thousands of category pages. Article pages are organized under category pages, wherein a single article page can belong to multiple categories. We argue that the entire Wikipedia can be thought of as a huge labeled document collection, with the article pages forming the documents, and the categories being the labels for the documents. Thus, in addition to the content of the pages, we can now use the categories as labels for training our embeddings. However, we still need to do some preprocessing on the raw wikipedia corpus to create our training dataset. We used a combination of JWPL[1] [22] and WikiExtractor[2] to process a recent dump (20170301) of English Wikipedia[3].

5.3.1 Category Processing

We removed the following categories from our Wikipedia dump:

1. Categories marked as hidden.
2. Categories containing a number in their name, for instance "Births in 1992".
3. Additionally, we removed all the categories containing either of the following in the name:

   _in_, _who_, _by_, _with_, Lists_of, Wikipedians, _having_, _on_, stubs, _of_, _from_, User, Wiki, articles, pages, categories, template, _counter, redirect

4. Finally, after removing the noisy categories using the filters from above, we removed all the categories that have less than 5 article pages assigned to them in the entire Wikipedia.

We are ultimately left with $189,385$ categories/topics, with $7,684,803$ occurrences in the entire Wikipedia.

[1] https://dkpro.github.io/dkpro-jwpl
[2] https://github.com/attardi/wikiextractor
[3] https://dumps.wikimedia.org/enwiki

5.3.2 Content Processing

We carried out the following steps in sequence for processing the content of the article pages:

1. We removed all Disambiguation, Redirect, Category, User, Template, Portal and Talk pages, leaving us with $5,174,808$ article pages.
2. We then tokenized these article pages using Spacy[4], removed all punctuations, numbers, and lowercased all tokens.
3. Then, we removed all pages with less than 100 tokens.
4. We then removed all pages containing *list_of* in the title.
5. We then removed all pages that have no category assigned to them from our selected categories from above.
6. We then removed the tokens that have a frequency of less than 7 in the entire Wikipedia.

We are ultimately left with over $2,536,585$ articles, containing over $1,430,117,083$ word occurrences, with a vocabulary of around $785,277$ words.

5.4 RELATED WORK

There have been several related attempts at modifying the Word2Vec skip-gram architecture to inject topics/concepts into the embeddings. We review a few of them below:

Li *et al*. [23] introduced two methods for learning Entity and Category Embeddings:

- **Category Embedding**: Here, target is a wikipedia entity page along with its categories, and context are the entities that are mentioned on that target wikipedia entity's page.
- **Hierarchical Category Embedding**: This method is similar to the one above, with the addition of weights to the categories of the entity page, with more weight for categories lower in the hierarchy, and less for the ones higher-up in the hierarchy.

Similarly, Yamada *et al*. [24] introduced two methods for learning Entity and Word Embeddings:

- **KB-Graph Embedding**: Here, target is the entity page, and the context are the other entity pages that are linking to that entity page.

[4]https://github.com/explosion/spaCy/

- **Context Embedding**: Here, target is the disambiguated entity mention, and the words and entities in its surrounding window make up the context. In addition, they add word-word predictions as in word2vec, and the KB-Graph model from above to the mix as well.

Likewise, Shalaby *et al*. [25] introduced two methods for learning Entity and Word Embeddings:

- **3C**: Here, target is the disambiguated entity mention, and the disambiguated entity mentions in the surrounding window make up the context.
- **CRC**: Here, target is a word, or a disambiguated entity mention, and context are the words or disambiguated entity mentions in the surrounding window. Note that this is very similar to the Entity2Vec embedding that we introduced earlier in our work.

Liu *et al*. [26] first ran LDA on the entire corpus, and then introduced 3 ways of learning topic sensitive word embeddings. In their system every latent topic and word has a separate embedding.

Hu *et al*. [27] considered an entity page as the target, and used other entities on that target entity's page as the context. They learn entity embeddings as well as the similarity metric. They harness signal from the Wikipedia Category hierarchy to learn a depth-sensitive similarity metric.

In stark contrast to the systems mentioned above, in our embeddings, target is the word appearing on an entity page, and the context encompasses the categories of that entity page in the Topic2Vec model, whereas the context encompasses the categories and the context words in the Word2Concept model.

5.5 TRAINING

We trained all embeddings for 3 epochs. We start with a learning rate of 0.5, and closely follow Mikolov *et al*.'s [3] method of linearly decaying it to zero by the end of the 3rd epoch. We use a maximum context window of 5 and uniformly sample a window from 1 to 5 during training. We also negatively sample 10 words to approximate the softmax, and subsample frequent words with a subsampling rate of 0.001. We use a batch size of 500, and use SGD for optimization. We use Tensorflow[5] for implementing all our models.[6].

Furthermore, since the objective function of the Word2Concept model involves 2 disjoint loss functions (i.e. one for the word-word interaction, and another for the word-topic interaction), we alternate between optimizing them, and use a λ of 1, i.e. equal contribution from both the word-word and word-topic losses.

[5]https://github.com/tensorflow/tensorflow/

[6]Our implementation is public at https://gitlab-beta.engr.illinois.edu/sgupta96/word2Concept

5.6 EXPERIMENTS & RESULTS

We carry out both extrinsic and intrinsic evaluation for our new embeddings. We use the word analogy task for intrinsic evaluation, and Dataless Classification for extrinsic evaluation.

5.6.1 Word Analogy

Following Mikolov *et al*. [3], we evaluate our trained embeddings on a word analogy task. The dataset consists of 8869 semantic and 10675 syntactic analogy questions. The task is to answer the questions of the type: "If Man is to Woman, then King is to ?". We follow the same procedure as Mikolov *et al*., and find the closest word embeddings to the embedding arithmetic of "Man - Woman + King", and evaluate against the actual answer of "Queen" using Precision@k as the metric. For our evaluation, we use Precision@1. Since we used a very different and smaller corpus as compared to the original work from Mikolov *et al*., our word2vec analogy numbers are not directly comparable to the published numbers. However, to ensure the correctness of our Tensorflow code, we obtained the word2vec embeddings from Song *et al*. [2], and compared their analogy performance with ours[7]. We report the evaluation results for the input word embeddings in table 5.1, and for the output word embeddings in table 5.2. We also report the numbers for the syntactic and semantic analogy splits of the dataset.

Dimension	Reported Word2Vec (syn | sem | total)	Word2Vec (syn | sem | total)	Topic2Vec (syn | sem | total)	Word2Concept (syn | sem | total)
100	33.6 | 51.1 | 41.5%	44.7 | 37.1 | 41.2%	25.8 | 23.5 | 24.8%	44.6 | 37.2 | 41.2%
300	N.A.	51 | 41.6 | 46.7%	28.1 | 24.9 | 26.6%	51.7 | 43.6 | 48%
500	51.7 | 72.7 | 61.2%	51.9 | 39.9 | 46.5%	28.2 | 25.4 | 26.9%	52.5 | 44 | 48.6%

Table 5.1: Word Analogy Precision@1 (Input Embeddings)

Dimension	Reported Word2Vec (syn | sem | total)	Word2Vec (syn | sem | total)	Topic2Vec (syn | sem | total)	Word2Concept (syn | sem | total)
100	33.6 | 51.1 | 41.5%	42.2 | 33.4 | 38.2%	25.8 | 23.5 | 24.8%	41.1 | 37.3 | 39.4%
300	N.A.	50.7 | 42.5 | 47%	28.1 | 24.9 | 26.6%	49.7 | 43.5 | 46.9%
500	51.7 | 72.7 | 61.2%	52 | 43.2 | 48.0%	28.2 | 25.4 | 26.9%	51.1 | 44.4 | 48%

Table 5.2: Word Analogy Precision@1 (Output Embeddings)

As can be seen from the tables, our trained word2vec embeddings are comparable to the word2vec embeddings from Song *et al*. on the analogy dataset when the number of dimensions is small.

[7] we couldn't get 300 dimensional word2vec embedding from the authors

However, interestingly, the gap between the embeddings widens as the number of dimensions go up. Moreover, we noticed that we are able to match the performance on the syntactic analogies, whereas, we do much worse on the semantic analogies. We attribute these observed patterns to the fact that we use much less training data (nearly half) as compared to the Song *et al*. Also, as expected, the performance of all embedding models increases with the increase in the dimensionality of the embeddings for a while, after which it saturates. Interestingly, we found the performance of the input word embeddings to be consistently superior to the output word embeddings.

We observed that our Topic2Vec embedding is quite inferior to the word2vec embedding on the analogy task. This can be attributed to the fact that the analogy dataset contains roughly 50% syntactic analogy questions, which Topic2Vec is not designed to capture.

Furthermore, we observed that our Word2Concept embedding marginally improves upon the Word2Vec embedding, however, we don't have a satisfactory explanation for this observation.

5.6.2 Dataless Classification

We also evaluated the usefulness of the new trained embeddings on a dataless classification benchmark. We used the embellished label descriptions from Song *et al*. [2], replaced the ESA word embeddings with our Topic2Vec and Word2Concept word embeddings, and used the same process of averaging word embeddings of all words in the document to get the document and label embeddings. We evaluated on the 20newsgroups dataset, using the bottom-up tree traversal for inference. We computed MicroF1 and MacroF1 using all the labels in the hierarchy. We report the Dataless numbers corresponding to the 100 dimensional embeddings in table 5.3, and for the 500 dimensional embeddings in 5.4. We report the numbers for both the input and output word embeddings (wherever applicable).

Performance	ESA (original | reduced)	Reported Word2Vec (original | reduced)	Our Word2Vec (input | output)	Topic2Vec	Word2Concept (input | output)
MicroF1	0.69 | 0.59	0.43 | 0.40	0.17 | 0.34	0.32	0.23 | 0.31
MacroF1	0.62 | 0.53	0.32 | 0.31	0.10 | 0.25	0.24	0.15 | 0.21

Table 5.3: Dataless Performance on 20newsgroups (100 dim.)

To our surprise, even though our word2vec embedding was comparable to the word2vec embedding from Song *et al*. [2] on the analogy dataset for 100 dimensions, we found it to be quite inferior

Performance	ESA (original | reduced)	Reported Word2Vec (original | reduced)	Our Word2Vec (input | output)	Topic2Vec	Word2Concept (input | output)
MicroF1	0.69 | 0.69	0.52 | 0.50	0.21 | 0.41	0.32	0.24 | 0.38
MacroF1	0.62 | 0.62	0.45 | 0.44	0.12 | 0.32	0.25	0.16 | 0.28

Table 5.4: Dataless Performance on 20newsgroups (500 dim.)

on the dataless benchmark, with our 100 dimensional word2vec embedding yielding a MicroF1 of 0.34 as compared to the reported MicroF1 of 0.43.

We then investigated if the poor performance on the Dataless Classification is a result of our usage of a much smaller vocab of 700K as compared to 3M in Song *et al*. We used the embeddings from Song *et al*. for only the words in our vocabulary, and noticed that the reported ESA and Word2Vec performances marginally decreased on our vocabulary, however, there was still a significant gap left between the performances. We present these results under the **"reduced"** sub-column in tables 5.3 and 5.4.

Moreover, we found the performance of the input word embeddings to be drastically inferior to the performance of the output word embeddings of word2vec, with the input embeddings yielding a MicroF1 of just 0.17 as compared to the 0.34 MicroF1 of the output embeddings. This further helps us conclude that an evaluation on the analogy dataset is not a good proxy for evaluating/optimizing embeddings for Dataless, since the performance of input embeddings was slightly superior to that of the output embeddings on the analogy dataset.

Contrary to our expectations, both Topic2Vec and Word2Concept embeddings perform inferior to our Word2Vec, which is way short of the much superior performance of ESA. To take first steps towards understanding the reason behind this observed behavior, we also created t-SNE plots [28] for some sampled word and category embeddings (more details in the appendix). Some preliminary analysis of the t-SNE plots (figures A.1, A.2, A.3 in the appendix) showed that all the word embeddings are doing a decent job at mapping similar words close to each other, however, pictorially we couldn't see many instances of the famous linearity in relations between the word embeddings. Also, our very preliminary analysis of the t-SNE plots showed that the topic embeddings are not doing a good job at mapping similar topics near to each other (figures A.4, A.5 in the appendix). We include all the t-SNE plots in the Appendix.

5.7 CHALLENGES

We faced many challenges while training these new embeddings. For instance, despite Word2Vec's popularity, we found that the impact of various hyperparameters on its performance is not very well understood in the literature. Thus, we struggled a lot while deciding the appropriate vocab size, number of categories, minimum length of pages etc. Moreover, since the training of the embeddings takes about a day, we found it impossible to experiment with a large number of hyperparameters on our modest GPU infrastructure, and had to rely on our instincts and published statistics to guide our hyperparameter choices. This problem was further compounded by the fact that Word2Vec is an unsupervised model, and thus apart from mimicking the number of epochs from the published work, we didn't have a better stopping criteria for our model training. Thus, we feel that our embeddings would have performed a lot better if our infrastructure had allowed a more elaborate hyperparameter sweep.

Also, despite the popularity of Word2Vec, we couldn't find decent Tensorflow implementations, and thus had to write the word2vec code from scratch with all of the tricks that are used in the original paper. This was another source for the introduction of noise in our evaluations.

CHAPTER 6: DISCUSSION & FUTURE WORK

In this chapter, we discuss some additional issues related to Dataless Classification, especially concerning the way it is evaluated currently. We also discuss some of the future work that can be done to improve Dataless Classification.

6.1 EVALUATION PROBLEMS

Dataless Classification is currently evaluated on existing text classification datasets like 20newsgroups and RCV1. However, these datasets were labeled in a discriminative manner, in which a document was given to the annotators, and they were asked to pick a category for the document from a predefined category set. Note that, the annotators were just tasked with differentiating one class from the another, and were not provided label descriptions that they were supposed to follow strictly and consistently while annotating the documents. Thus, these datasets don't provide any label description that the Dataless Classifier can use. In the absence of label descriptions from the source, the onus of coming up with label descriptions lies on the researchers evaluating Dataless Classification, as in Song *et al*. However, the researchers are often not familiar with the nuances of the target collection, the contents of the documents, and the guidelines that were used for labeling the documents, and thus can't be expected to be able to come up with label descriptions that can do a good job at separating the labels. This is clearly evident from the confusion matrix for ESA based dataless classification, in which documents with the gold label "Sales" were mis-classified under "Computer" 60% of the times, and under "Automobiles" 13% of the times. Intuitively, in the absence of any label description, this mis-classification makes sense since "Sales" and "Computers", or "Sales" and "Automobiles" are very related classes, and it is possible that the researchers were unable to capture the dataset-specific differences between those labels through their label descriptions. Similarly, documents with the gold label "Electronics" were mis-classified under "Computer Hardware" 36% of the times, and under "Computer Graphics" 8% of the times.

One of the ideas that we tried to address this problem was to use some of the labeled documents themselves as the label description. In such a formulation, the researchers experimenting with Dataless Classification can afford to be completely oblivious of the contents of the documents or the meaning of the labels. Furthermore, in this formulation, Dataless Classification can be thought of as a One-Shot Learning or a K-Shot learning method. We carried out two preliminary experiments towards this goal. In one formulation, we selected the longest document in each label as the label description, and in the other, we randomly sampled 5 documents from each label and averaged their representations to get the label representations. We report the corresponding

numbers in Table 6.1.

Label Desc.	Manual	Longest Doc.	Averaged Doc.
Micro-F1	0.69	0.56	0.40
Macro-F1	0.62	0.46	0.31

Table 6.1: Documents as Label Description

As can be seen in the table, the averaged document representation performs inferior to the longest document based representation, which in turn performs inferior to the original keyword based representation. This is easy to explain, since a single labeled document or even an average over 5 documents can't possible capture all the dataset-specific nuances to give us a very rich label representation. This is further confirmed by the significant difference between the performances of the longest document based label representation and the averaged document based label representation, which shows the variability in the contents of the documents and thus in the label representations too.

6.1.1 New Evaluation Dataset

As is clear from the issues outlined above, in order to support future dataless classification research, it is crucial to create some new datasets that are labeled with a consistent label description, and that such label descriptions are made available as well. Towards this goal, we propose a way to use existing resources for creating such a dataset.

Wikipedia contains millions of articles and thousands of categories, wherein the articles are classified under the said categories. Not only is Wikipedia huge in size, but it is also well-curated, containing articles of high quality, and fairly consistent organization of articles under the categories. Moreover, the categories are naturally organized in a hierarchy, and some of them even have corresponding article pages. For instance, the category "Politics" which has hundreds of pages assigned to it, also has a corresponding article page titled "Politics" that explains Politics. Thus we propose the following method for creating a dataset for dataless classification using Wikipedia:

1. Select all the category pages that have a corresponding article page as well. This constitutes the super-set of the categories that we can work with. We found $\approx 200,000$ categories with a corresponding article page in our Wikipedia dump.

2. Keep N categories from all such categories, where we can choose to keep only the top-level categories with a large number of article pages assigned to them. This will constitute the

target label set for the dataset.

3. Select all the articles that are organized under any of these N categories. This constitutes our labeled document set. We can now divide this labeled set into Train, Dev, and Test corpora.

It will be interesting to try out all the new embeddings introduced in this work on this new dataset in the future.

6.2 LEARNING COMPOSITIONS

We had identified 6 limitations of ESA based document representations in section 2.5. However, we only attacked, and proposed solutions for 5 of those limitations in this work. However, the 6^{th} limitation of *Compositionality* is central to all the ideas that have been tried out in this work, and thus could be impacting their performance as well. Specifically, we've been using a weighted-average of word or entity representations to form the document representations until now. However, languages exhibit non-compositionality, wherein a phrase's or a sentence's meaning is not always derivable from its constituent words. Moreover, the order of words plays a major role in some languages, whereas the document representations that we've been creating until now have been agnostic to the order. Thus, it begs the question if it is possible to learn a suitable composition function to create more useful document representations.

Supervised techniques that use positional features or that use models like LSTM, inherently learn to compose the word representations to form the document representation in a task-specific manner. However, since we don't expect any labeled documents in the Dataless Classification formulation, it is not trivial to learn a task-specific composition function.

Towards this goal, we propose a Distant Supervision based idea for learning useful document representations. The success of our idea hinges on the availability of a special labeled resource. We expect the said resource to have the following properties:

- The resource is labeled with a large number of general-purpose categories.
- Most concepts in the world can be differentiated with each other using the categories from this resource.
- The resource contains a large number of documents labeled with the above categories.

If we have such a resource available, we propose the following for carrying out Dataless Classification in which the composition function or the document embeddings are learnt instead of relying on a heuristic (such as "weighted-average"):

1. Train a neural document classification model on this labeled resource.

2. Now, the target document and the label description in the dataless classification task can be fed to this network to produce the document and label representations. The representations can either be the one-hot embedding of the predicted labels, or it can be an intermediate representation from the network.

3. Now, similar to Dataless, these document and label representations can be used with a similarity metric such as cosine for classification.

As it turns out, we already have such a resource available – Wikipedia. Wikipedia contains millions of articles, which will form the document part of this collection, and Wikipedia's thousands of categories will form the label part of the document. Furthermore, since Wikipedia is vast and generic, its categories can be considered to be diverse enough to be able to differentiate between any 2 known concepts. For instance, if Wikipedia has categories like "Politics" and "Environment", then this idea can be used to differentiate between two new labels such as "Climate Change" and "Non-Vegetarianism". The idea here is that the two labels will probably have equal weights assigned to the "Environment" dimension, but will be distinguishable using the "Politics" dimension.

We can use the preprocessing steps mentioned in section 5.3 for creating a Wikipedia based training dataset. We can then use this dataset for learning any deep architecture. For instance, similar to Kim *et al*. [29] and Johnson *et al*. [30], one can use CNNs for document classification using words as one-hot vectors, or like Zhang *et al*. [31] one can apply a character-level CNN. Or like Tai *et al*. [32] one can choose to model the structure of the sentence using a tree-structured LSTMs. Similar to Lai *et al*. [33] and Zhou *et al*. [34], combining LSTM and CNN structure is also an option. It will be interesting to benchmark the performance of some of these ideas against ESA in the future.

CHAPTER 7: CONCLUSION

In this work we identified some key limitations of Dataless Classification, more specifically, of the underlying ESA embedding. We then developed techniques for addressing those limitations, and in the process introduced several new embeddings.

EntityESA and Entity2Vec target the ambiguity or polysemy problem of ESA by disambiguating words, thus leading to cleaner representations. However, they fail to deliver gains because of the propagation of errors introduced by the underlying Entity Linking system.

Random Projection and densification using other dense or sparse embeddings methods target the computationally expensive problem of ESA. We found both Random Projection, and densification using EntityESA methods to be promising if the intention is to create a dense version of ESA which is able to keep some of the benefits of the ESA, and performs much better computationally.

Topic2Vec and Word2Concept on the other hand target the sparsity problem of ESA by learning to embed topics in a dense representation. However, these embeddings failed to deliver gains on top of ESA as well.

We then argued that some of these observations are because of the underlying dataset that is being used for evaluation – since the said dataset was never labeled using a consistent label definition in mind. We then proposed a way to create a new dataset that can be used for future dataless classification evaluations.

We then finally proposed a method that can be used to address the compositionality limitation of ESA. This method should be evaluated in the future.

APPENDIX A: EMBEDDINGS VISUALIZATION

A.1 T-SNE PLOTS

This appendix contains the t-SNE plots [28] for the 100 dimensional word and topic embeddings (projected to 2D) for each of the embedding models. For the word embeddings, we selected 137 words from the total 905 words in the analogy dataset, and selected 99 representative categories from over 190k categories for visualizing the category embeddings. The t-SNE plots have been manually annotated (wherever possible) to aid the visualization and put emphasis on certain patterns, however, the annotations should be considered with a grain of salt since they are quite preliminary and have been cherry-picked (P.S. the plots have not been optimized for print).

Figure A.1: t-SNE plot for Word2vec (100 dims.)

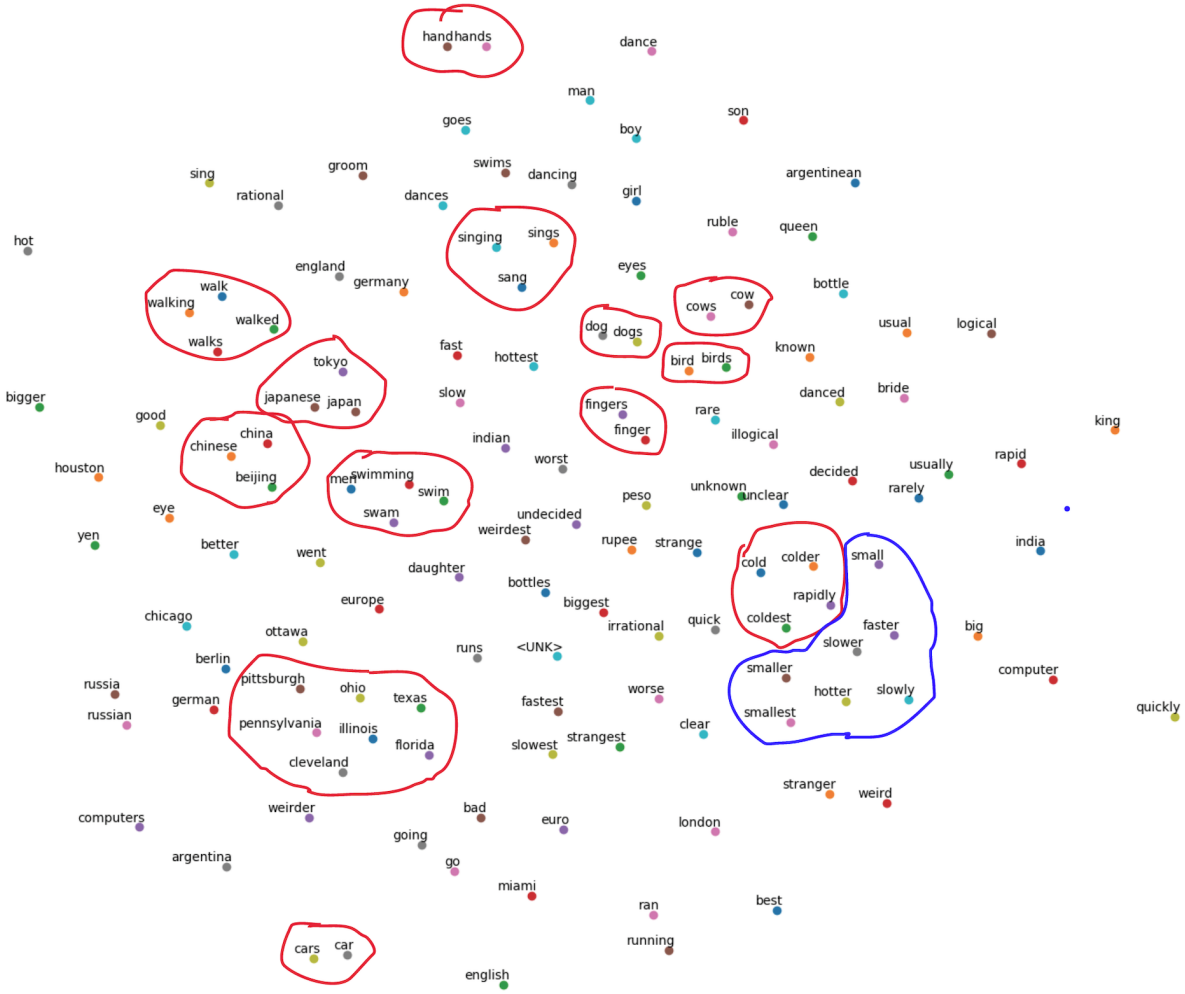

Figure A.2: t-SNE plot for Topic2Vec word embeddings (100 dims.)

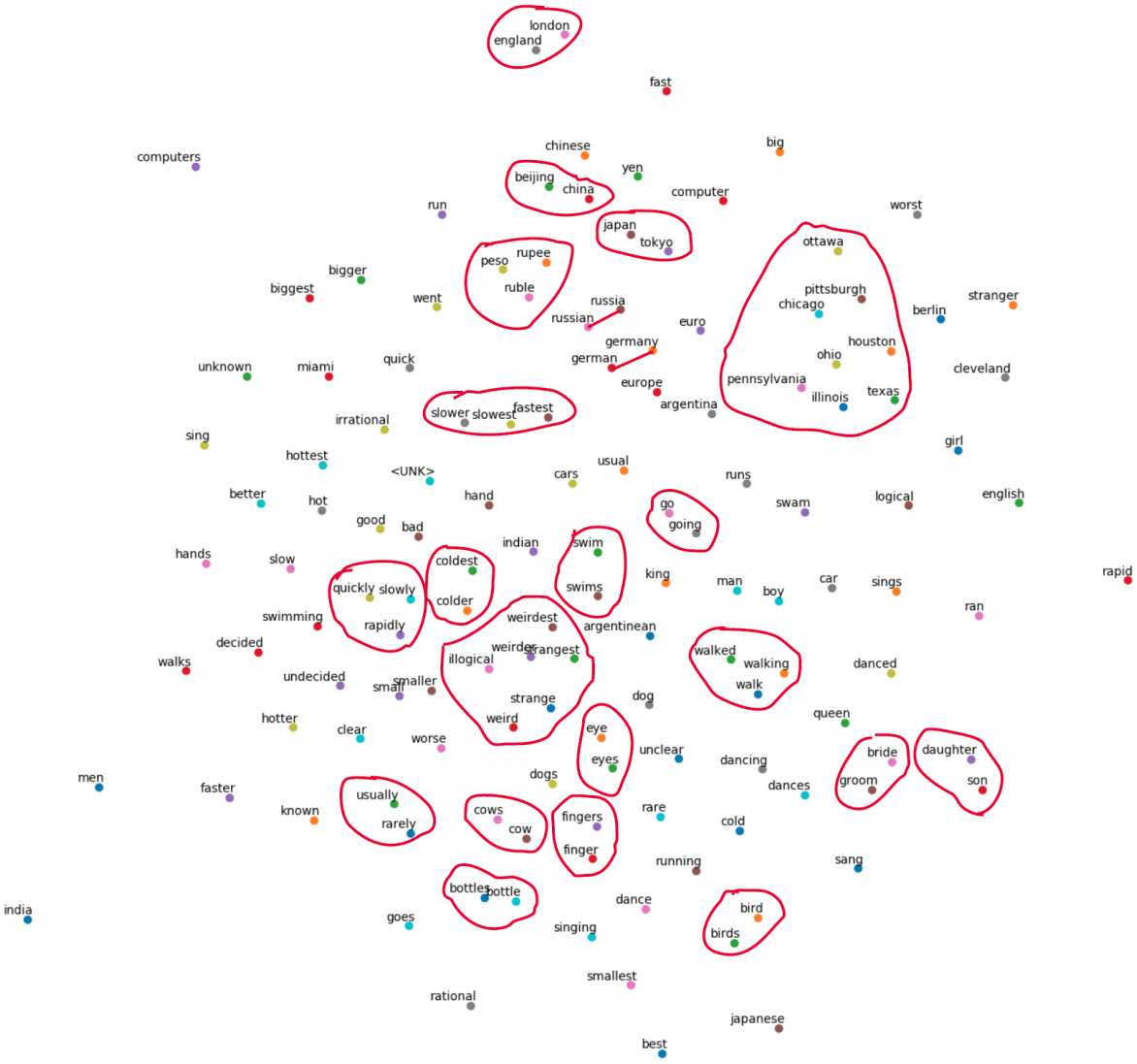

Figure A.3: t-SNE plot for Word2Concept word embeddings (100 dims.)

Figure A.4: t-SNE plot for Topic2Vec topic embeddings (100 dims.)

Figure A.5: t-SNE plot for Word2Concept topic embeddings (100 dims.)

REFERENCES

[1] M.-W. Chang, L.-A. Ratinov, D. Roth, and V. Srikumar, "Importance of semantic representation: Dataless classification." in *AAAI*, 2008, pp. 830–835. 2, 6, 10, 11

[2] Y. Song and D. Roth, "On dataless hierarchical text classification," in *Proceedings of AAAI*, vol. 14, 2014, pp. 1579–1585. 2, 6, 13, 27, 33, 34

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013. 3, 27, 32, 33

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119. 3

[5] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis." in *IJCAI*, vol. 7, 2007, pp. 1606–1611. 4

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003. 5

[7] T. Hofmann, "Probabilistic latent semantic analysis," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 289–296. 5

[8] Y. Yang, D. Downey, J. Boyd-Graber, and J. B. Graber, "Efficient methods for incorporating knowledge into topic models," in *Empirical Methods in Natural Language Processing*. 5

[9] D. Ramage, C. D. Manning, and S. Dumais, "Partially labeled topic models for interpretable text mining," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 457–465. 5

[10] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, 2009, pp. 248–256. 5

[11] D. Andrzejewski, X. Zhu, and M. Craven, "Incorporating domain knowledge into topic modeling via dirichlet forest priors," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 25–32. 5

[12] D. Andrzejewski, X. Zhu, M. Craven, and B. Recht, "A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1171. 5

[13] D. Andrzejewski and X. Zhu, "Latent dirichlet allocation with topic-in-set knowledge," in *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*. Association for Computational Linguistics, 2009, pp. 43–48. 5

[14] L. Ratinov, D. Roth, D. Downey, and M. Anderson, "Local and global algorithms for disambiguation to wikipedia," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 1375–1384. 8

[15] D. Milne and I. H. Witten, "Learning to link with wikipedia," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 509–518. 8

[16] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "Dbpedia spotlight: shedding light on the web of documents," in *Proceedings of the 7th International Conference on Semantic Systems*. ACM, 2011, pp. 1–8. 8

[17] P. Ferragina and U. Scaiella, "Tagme: on-the-fly annotation of short text fragments (by wikipedia entities)," in *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010, pp. 1625–1628. 8

[18] D. B. Nguyen, J. Hoffart, M. Theobald, and G. Weikum, "Aida-light: High-throughput named-entity disambiguation," *Linked Data on the Web at WWW2014*, 2014. 8

[19] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani, "Dexter: an open source framework for entity linking," in *Proceedings of the sixth international workshop on Exploiting semantic annotations in information retrieval*. ACM, 2013, pp. 17–20. 8

[20] X. Cheng and D. Roth, "Relational inference for wikification," *Urbana*, vol. 51, p. 61801, 2013. 8, 14

[21] D. Khashabi, M. Sammons, B. Zhou, T. Redman, C. Christodoulopoulos, V. Srikumar, N. Rizzolo, L. Ratinov, G. Luo, Q. Do et al., "Cogcompnlp: Your swiss army knife for nlp," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018. 14

[22] T. Zesch, C. Müller, and I. Gurevych, "Extracting lexical semantic knowledge from wikipedia and wiktionary." in *LREC*, vol. 8, no. 2008, 2008, pp. 1646–1652. 30

[23] Y. Li, R. Zheng, T. Tian, Z. Hu, R. Iyer, and K. Sycara, "Joint embedding of hierarchical categories and entities for concept categorization and dataless classification," *arXiv preprint arXiv:1607.07956*, 2016. 31

[24] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, "Joint learning of the embedding of words and entities for named entity disambiguation," *arXiv preprint arXiv:1601.01343*, 2016. 31

[25] W. Shalaby and W. Zadrozny, "Learning concept embeddings for efficient bag-of-concepts densification," *arXiv preprint arXiv:1702.03342*, 2017. 32

[26] Y. Liu, Z. Liu, T.-S. Chua, and M. Sun, "Topical word embeddings." in *AAAI*, 2015, pp. 2418–2424. 32

[27] Z. Hu, P. Huang, Y. Deng, Y. Gao, and E. P. Xing, "Entity hierarchy embedding." in *ACL (1)*, 2015, pp. 1292–1300. 32

[28] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008. 35, 42

[29] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014. 40

[30] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *arXiv preprint arXiv:1412.1058*, 2014. 40

[31] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657. 40

[32] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015. 40

[33] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." in *AAAI*, 2015, pp. 2267–2273. 40

[34] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015. 40