From Context Embeddings to Meaning Embeddings

David Yenicelik
ETH Zürich

A dissertation submitted to ETH Zürich
in partial fulfilment of the requirements for the degree of
Master of Science ETH in Computer Science

Under the supervision of
Florian Schmidt
Yannic Kilcher
Prof. Dr. Thomas Hofmann

Eidgenössische Technische Hochschule Zürich
Data Analytics Lab
Institute of Machine Learning
CAB F 42.1, Universitätsstrasse 6, 8006, Zürich
Zürich 8006
Switzerland

Email: yedavid@ethz.ch

April 11, 2020

Declaration

I David Yenicelik of ETH Zürich, being a candidate for the M.Sc. ETH in Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 14,235

Signed:

Date:

Abstract

Write a summary of the whole thing. Make sure it fits in one page.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In Natural Language Processing (NLP), bilingual lexical induction (BLI) is a problem of inferring word-to-word mapping between two languages. While supervised BLI may be learned trivially from a dictionary, unsupervised BLI is highly non-trivial, and serves as a backbone to many unsupervised Neural Machine Translation (NMT) systems, without which the overall MT performance drastically drops [**?**] [**?**]. In addition to the unsupervised setting, words in a source language A often do not carry a one-to-one correspondence with words in a target language B. This further increases the difficulty of finding a (bijective) mapping between the two embedding spaces.

1.1 Scope of Work

Continuing where [**?**] left off, we want to investigate the performance of using normalising flows to model unsupervised lexicon matching between two probability distributions, which are defined by Gaussian emebddings, which have a one-to-one correspondence to tokens in the respective languages. We aim to use Gaussian embeddings for the robustness, and better integration into the probabilistic perspective. The method discussed in the paper also relies a lot on the (semi-)supervised loss component. We aim to investigate why this is the case, and would like to revise a method which is more robust

in the fully unsupervised setting.

Minimal goals: We propose the following steps on achieving this goal.

1. Replicate the algorithms and models which can generate through Gaussian embedding [**?**]. Do one sanity check by doing a sanity check on one of (SimLex / WordSim)

2. Replicate "Density matching for bilingual word embedding" to setup a baseline normalising flow between vector-word-embeddings [**?**].

3. Define loss between predicted and target embeddings (if change in definition is necessary)

4. Change the embeddings in point 2. to use Gaussian embeddings. Implement Loss functions found in point 3.

Extended goals: If the above points provide good performance , we would like to expand on the below points.

1. Implement a fully unsupervised extension by investigating the shortcomings of [**?**].

2. Investigate "deeper" normalising flows than the linear flow in [**?**] as such [**?**] [**?**].

Contingency Plan / Further extended goals: Implementing the following points would allow for an applied perspective of this approach, showing that this methodology allows for more robust mapping, also outside the field of NLP.

1. Train embeddings for job-systems ESCO and AUGOV using Skip-Gram or Co-Occurence-matrix based. Do a sanity check.

2. Train Gaussian embeddings for job-systems. Do a sanity check.

3. Generate a small validation dataset between the European- and Australian job-system.

4. Setup some baseline algorithms based on NLP, graph-matching, colinear-PCA for matching as a non-NLP benchmark environment. Compare against above-proposed methods.

5. Find a normlising flow model to transform one job system into another.

TODO: Fill out at the very end!

Chapter 2

Motivation

- Natural Language Understanding (NLU) lies in the intersection of formalising human text into a computer-readable format. - Computer-readble units must be numerical, thus we represent words and meanings by vectors - The relationships between vectors should cover underlying relations between the word meaning - Language models are a generalization of word vectors - Word embeddings such as Word2Vec and language models such as BERT are used for other tasks, which are refered to as "downstream" tasks, due to their nature of using up these word-embeddings and language models

- Because these are the most basic units of text, any shortcomings and properties will propagate over to any downstream task - Some properties include that static word vectors like word2vec form a bijection between discrete vectors and word tokens. However, because a single word can entail multiple meanings, such as the polysemous word "bank" ((1) financial instution, (2) a sitting bench), this results in a lossy compression - Other language models like context embeddings entail too much information, and also include other linguistic features such as semantic information, relatenedness to unrelated concepts. These properties can easily introduce bias into any downstream tasks. - We conjecture that many language tasks incl. translation will benefit most from meaning information

- Our general approach is to start with a complex language model that outputs context embeddings, and find signals / vectors that entail meaning.
- Although this work is dedicated to the domain of natural language understanding, the principles analysed in this work should generalize to other domains with similar structural properties as well, where we want to denoise some embedding space to some select properties.

Chapter 3

Background

TODO: Take some more from here

- Some of the main points behind [**?**] are that there is inherent structure in language, and that this structure can be formalized. - [**?**] mentions that the relation between the linguistic representation in terms of sounds and tokens are related to the meaning that the representation entail. - However, despite the obvious relationship, the distinction between distributional structure of the language and meaning is not always clear. and that there is a "parallel" meaning structure, and argues that there is not a one-to-one relation between vocabulary and classification of meaning. - Also argues that the meaning of a word is entailed by it's environment, and the words that it occurs with. - Other viewpoints (CITE HOFMANNs "teachers") In the end, language captures the evolution of human thought.

- cite paper "meaning is classified by its context"

Chronologically, the following data structures are manifestations of the above idea of defining a word by it's neighborhood.

3.1 Word Embeddings

In general, we want to find a mapping

$$w^t \mapsto w \in \mathbb{R}^d \tag{3.1}$$

where w^t is a token representation and where this token representation is transformed into a d-dimensional vector representation.

One of the early ideas of representing the distributional structure of words through a data structure was expressed in [**?**].

[**?**] argues that a statistical model of language can be represented by the conditional probability of the next word given all the previous ones.

$$\hat{P}\left(w_1^T\right) = \prod_{t=1}^{T} \hat{P}\left(w_t | w_1^{t-1}\right) \tag{3.2}$$

This is then relaxed using the markovian assumption that the future state is independent from the previous states given the current state. Using this assumption, we arrive at the relaxed formulation of

$$\hat{P}\left(w_t | w_1^{t-1}\right) \approx \hat{P}\left(w_t | w_{t-n+1}^{t-1}\right) \tag{3.3}$$

Another approach is to take a more global approach, and consider not only the preivous $t-1$ words, but the full context / neighrborhood of a word

$$\hat{P}\left(w_1^T\right) = \prod_{t=1}^{T} \hat{P}\left(w_t | w_1^{t-1}\right) \tag{3.4}$$

We want to use some rank-based Energy, not absolute energy, such that we can make sure that neighboring words are pushed together, and two "random" words are pushed further away from each other.

The above two problems can be regarded as finding an invertible transformation from one (embedding) space $\mathcal{X} \in \mathbf{R}^d$ into another $\hat{\mathcal{X}} \in \mathbf{R}^d$. Normalising

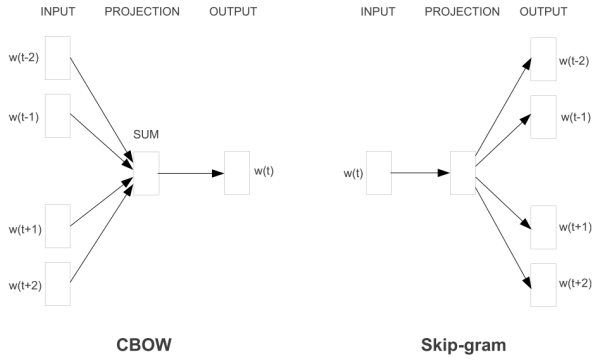

Figure 3.1: Figure taken from [**?**]. The CBOW architecture predicts the current word based on the context. The Skip-gram predicts surrounding words given the current word.

flows [**?**] [**?**] have proven to be a powerful tool in modeling such relations. As such, the aim of this project is to find a model based on normalising flows which is able to find an an invertible mapping f from $\mathcal{X}$ to $\hat{\mathcal{X}}$. To keep the discussion focused, this thesis deals with the problem of finding a model for bilingual lexicon matching.

3.1.1 Static Word Embeddings

Static word embeddings are structural representations of words that for which each word token is represented by a vector, the so called *word vector*.

Word2Vec

The most prominent example of word vectors manifests itself in the work of [**?**]. Here, a neural network with a single embedding layer can be trained to transform one-hot-vectors into a latent vector representation $w \in \mathbf{R}$.

The training procedure follows a *continuous bag of words* methodology, or a *continuous skip gram model*. From experience, the skip gram model seems to be prevalent in the data science community.

Intuitively, this model is able to very well capture the idea of [**?**] that the meaning of a word is captured by its neighborhood.

The underlying principle of [**?**] is employed by the assumption that not only previous words, but the entire context of the word is taken into consideration when such a word vector is calculated.

However, other work argues that the context of a word is determined by its entire context. In contrast to (3.1), the underlying assumption can be represented as

$$\hat{P}\left(w_1^T\right) = \prod_{t=1}^{T} \hat{P}\left(w_t | w_1^{t-1}\right) \tag{3.5}$$

GLoVe

Gaussian Embeddings

Gaussian Embeddings have been first proposed in the context of words, although prior work has been adapted in embedding matrix-rows into a mean and standard deviation [**?**]

It is a continuous probabilistic relaxation of the otherwise common discrete point vectors. Each word is represented by a Gaussian distribution in high-dimensional space, allowing to better capture uncertainty and a representation and it's relationships. It can also express asymmetric relations more naturally than dot-products or cosine similarity, and enables for better-parametrized rule between decision boundaries.

Fitting Gaussian Mixture Models on embeddings have been done in order to apply Fisher kernels to entire documents.

Because this is an unsupervised learning task, we must use an energy function which is incorporated within a loss function that we try to minimize. The energy function describes (dis-)similarity between two items. The authors propose the following energy functions to derive a Gaussian word-embedding.

$$L_m(w, c_p, c_n) = max(0, m - E_\theta(w, c_p) + E_\theta(w, c_n) \tag{3.6}$$

Here, w is the word we want to sample, c_p is a "positive" context word, i.e. a word that co-occurs with the word w, and c_n is a "negative" context word, i.e. a word that does not co-occur with the word w. Usually the negative context words is sampled randomly from the corpus. The loss function reminds of a hinge-loss in logistic regression.

The authors propose two possible ways to learn the mean and variance of the Gaussian embeddings. They argue that the empirical covariance is not the most effective method of deriving the words as Gaussian embeddings. This does not allow for inclusion between ellipsoids

Symmetric similarity: expected likelihood or probability product kernel We can use any kernel (which is symmetric by definition) to derive at an energy function. For two Gaussians $f(x)$, $g(x)$, the inner product is defined as:

$$E_\theta(w, c) = \int_{x \in \mathcal{R}^d} f(x)g(x)dx \tag{3.7}$$

$$= \int_{x \in \mathcal{R}^d} \mathcal{N}(x; \mu_w, \Sigma_w)\mathcal{N}(x; \mu_c, \Sigma_c)dx \tag{3.8}$$

$$= \mathcal{N}(0; \mu_w - \mu_c, \Sigma_w + \Sigma_c) \tag{3.9}$$

For numerical feasibility and easy of differentiation, we usually maximize the $\log E_\theta(w, c)$ for a given dataset with $w \in \mathcal{W}, c \in \mathcal{C}$. We will not go further in what the specific gradient of this log-energy is.

Asymmetric divergence: KL-Divergence We can use more directional supervision to exploit directional supervision, such as a knowledge graph.

Following energy-function is optimized:

$$-E(w_i, c_j) = D_{KL}(c_j||w_i) \tag{3.10}$$

$$= \int_{x \in \mathcal{R}^d} \mathcal{N}(x; \mu_{w_i}, \Sigma_{w_i}) \log \frac{\mathcal{N}(x; \mu_{c_j}, \Sigma_{c_j})}{\mathcal{N}(x; \mu_{w_i}, \Sigma_{w_i})} dx \tag{3.11}$$

$$= \frac{1}{2} \left(\mathrm{tr}\left(\Sigma_i^{-1} \Sigma_j\right) + (\mu_i - \mu_j)^\top \Sigma_i^{-1} (\mu_i - \mu_j) - d - \log \frac{\det(\Sigma_j)}{\det(\Sigma_i)} \right) \tag{3.12}$$

Because of the loss function, this can entail information such as "y entails x" as a soft form of inclusion between two datasets (if KL divergence is used). If a symmetric loss function is used, then this would most likely lead to overlap (IS THIS TRUE...???)

Uncertainty calculation: In contrast to the empirical standard deviation as an uncertainty measure, we can now calculate the uncertainty of the inner product (i.e. the distribution $P(z = x^T y)$ using the following formula

$$\mu_z = \mu_x^T \mu_y \Sigma_z = \mu_x^T \Sigma_x \mu_x + \mu_y^T \Sigma_y \mu_y + \mathrm{tr}(\Sigma_x \Sigma_y) \tag{3.13}$$

We then get an uncertainty bound, where c denotes the number of standard deviations away from the mean.

$$\mu_x^\top \mu_y \pm c \sqrt{\mu_x^\top \Sigma_x \mu_x + \mu_y^\top \Sigma_y \mu_y + \mathrm{tr}(\Sigma_x \Sigma_y)} \tag{3.14}$$

We can learn the parameters Σ and μ for each of these embeddings using a simple gradient-based approach, where we set hard constraints on

$$\|\mu_i\|_2 \leq C, \forall i \tag{3.15}$$

$$mI < \Sigma_i < MI \tag{3.16}$$

The method shows competitive scores to the Skip-Gram model, although usually only with minor improvements depending on the benchmark-dataset.

3.1.2 Context Embeddings

The Transformer Architecture

All of the below presented models use the transformer architecture (cite the paper "attention is all you need!")

ELMo

BERT

GPT and GPT-2

3.1.3 Other methods

Although the above presented methods are the prevailent methods for word-embeddings, there are also other methods which do not clearly fit into one of the above categories.

Generating "static" word-embeddings through contextual embeddings

Some work has been done in extracting word-embeddings from contextual language models like BERT or ELMo.

CITE (BERT WEARS GLOVES: DISTILLING STATIC EMBEDDINGS FROM PRETRAINED CONTEXTUAL REPRESENTATIONS)

(1) Uses *pooling* between BERT tokens to arrive at a single representation between words.

Here, sentences are split by space (tokenized). Words are tokenized further into a subword as defined by WordPiece (Wu et al. 2016).

The defined pooling operations looks as follows to arrive at the word from the individual subwords:

$$\mathbf{w}_c = f\left(\mathbf{w}_c^1, \ldots, \mathbf{w}_c^k\right); f \in \{\min, \max,\ \text{mean},\ \text{last}\ \}$$

where we have subwords $w^1, \ldots, w^k$ such that cat $\left(w^1, \ldots, w^k\right) = w$

Why would any of these pooling operations result in a meaninigful source-word? This is just squishing tokens together!

-¿ This is a major limitation for which we may need to use ELMo -¿ However this may be needed for "unseen concepts" (which are unseen words...) -¿ Perhaps check what fasttext does...?

(2) Uses *context combination* to map from different contexts $c_1, \ldots, c_n$ to a single static embedding w that is agnostic of context.

Proposed are two ways to represent context.

Decontextualization For a single word-context, we siimply feed-in the word by itself to the model.

Aggregated combine w in multiple contexts. n sentences are sampled from the dictionary $\mathcal{D}$. From the multiple sampled words, we then apply pooling to arrive at a single representation that aggregates the different tokens into one.

$$\mathbf{w} = g\left(\mathbf{w}_{c_1}, \ldots, \mathbf{w}_{c_n}\right); g \in \{\min, \max,\ \text{mean}\ \}$$

This post extracts (token?) word-embeddings: (https://towardsdatascience.com/nlp-extract-contextualized-word-embeddings-from-bert-keras-tf-67ef29f60a7b)

This seems to be a way to extract embeddings for tokens from BERT (https://github.com/img
embedding)

(-¿How can we create a (parametric) probability density from a point-cloud distribution?)

perhaps not necessarily interpretable in standard euclidean space (https://www.kdnuggets.co

features-interbertible.html) original (https://medium.com/thelocalminima/are-bert-features-interbertible-250a91eb9dc)

Perhaps we can mask all but the target token to arrive at one vector per token (and then combine them somehow...). But how do they extract the singular word-embeddings...?

(-¿ you could be like "acquiring bedeutung" is a big problem in many tasks. especially useful when we try to map one concept to another. we look at the NLP task for concreteness)

Generally, really good critique on this paper:

(https://openreview.net/forum?id=SJg3T2EFvr)

usually, we have sentence-embeddings, and do not look at word-embeddings.

(-¿ we don't want to add more and more context. we want a model which contains the polysemy of different contexts, which could allow for probability maximization..., otherwise we have to look at bigger and bigger documents to build more accurate language models, which becomes infeasible at some point. (although this would be the way humans work, because they live in context as well)

This blog aims to generate word-embeddings (and sentence-embeddings) from the BERT model. (https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/)

create word-vectors by taking out what BERT predicts at the nth token. create word-vectors by concatenating or summing multiple layer's outputs.

the cosine similarity between these vectors seem pretty well-done!

(-¿ Does it make sense to use BERT and then on calculate word-embeddings through an extended fully-connected model)

-¿ ELMo may provide a better tokenizer, maybe better ot use this? What about GPT? ELMo uses moses tokenizer which seems word-level enough

-¿ How to solve this tokenization problem....

-¿ Can also analyze only words that exist.

Chapter 4

Related Work

4.1 Structure inside BERT

(How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings)

going down the drain of "geometry" of BERT and ELMo.

could also go down the drain of bias (we would prefer to have uniform space over gender etc.)

-¿ does projection into some subspace which has same metric properties perhaps not make it asitropic?

pretty ok summary of what kind of properties we want from word-embeddings... (https://devopedia.org/word-embedding)

Especially in Named Entity Recognition (NER), there is a lot of use for static word-embeddings. I guess this is because we need static embeddings which represent the individual clusters?

-¿ Using pooling for some

-¿ Character level operation

-¿ Perhaps make good sense to work towards a word-embeddings where different vectors are close to each other?

-¿ perhaps find a metric space warping the vectors, s.t. an isotropic representation is achieved?

-¿ Perhaps tokenization is a big problem, but perhaps other architecture..? but retraining is too difficult.. probably best to just stick to BERT? one way or the other, we need good word-embeddings derived from good language models to form a probabilistic prediction of the concept

-¿ Could perhaps also try to make an adversarial autoencoder after the BERT layer (or continue training, s.t. a second loss is mimized as a downstream task?)

-¿ Perhaps distilling with "correct" tokens? i.e. another network which copies BERT, but instead of outputting ##end, it outputs one of most frequent 20k words

-¿ thesaurus using a (set of) words. a little like sentence-generation, but generating most-probable examples

-¿ Everyone just averages token-embeddings..

-¿ perhaps fitting a GMM to the contextualized representations of BERT may give a good probability space..?

-¿ Perhaps make sense to apply MUSE to this?

-¿ Artetxe 2019 uses language models to generate embeddings. we also do this, but do it using 1) better language models, and 2) better

-¿ QUESTION: Which factors (mapping algo, embedding) is delimiting in automated embedding matching

-¿ perhaps create a GMM for each concept, based on how many modals we identify? how to estimate the number of clusters? by graph-clustering perhaps! (this could be very consuming)

-¿ Adversarial autoencoder on BERTs last models to enforce it to some better

distribution

4.2 Metric Learning and Disentanglement

4.3 Zero shot and One shot learning

4.4 Clustering Algorithms

4.5 Applications of word vector

Word2Vec

BERT conditions on the rest of the input sentence.

BERT uses words, subwords and individual characters (in total 30'000) that are then used as tokens.

Idea is to do the following: Concepts (and thus words), are represented across multiple contexts. We can create probabilistic word-embeddings by sampling from a corpus the context of a specific word. From multiple samples of the context-embedding-vector, we take the mean and stddev, or create a GMM if there are multimodal logic (we can check this multimodality by runniing some sort of rejection sampling algorithm). Then we have a probability density function (one for each language), and can map one into another.

Perhaps we could split up too high-variance embeddings to multimodal embeddings, depending on their use-cases.

This allows for interpretability in polysemy sentences.

Not using more complex flows implies that the flow itself is not the bottleneck (they probably tried out other flows as well).

Are the individual word-embeddings going to be one big blob with high variance, or is it going to be a multi-modal distribution...?

Another task we may look at is, from a given word-embedding, sample it's context. Not entirely sure how to do this with BERT and co.

At what point do we overfit, and at what point do we generalize?

Artetxe bilingual token matching through unsupervised machine translation

- Input is cross-lignual word-embeddings - Build an unsupervised phrase-based statistical machine translation system - Derive phrase-based embeddings from input-word-embeddings by taking top 400'000 bigrams and 400'000 trigrams -¿ take arithmetic mean of word-embedding - score top 100 close phrases using softmax cosine similarity - generation of synthetic parallel cor-

pus using this approach - Then use FastAlign to use parallel corpus to align words -

Chapter 5

Related Work

As outlined by (https://ruder.io/cross-lingual-embeddings/) the task of finding cross-lingual embedding models fall within one of the four categories:

1. Monolingual mapping: Here we train the embedding for each language separately, and the try to find a mapping from one space into another. The learned mapping is usually linear.
2. Pseudo-cross-lingual: Here, a pseudo-language is created by mixing corpus from different languages. The cross-lingual context is used to learn the shared embedding space. [To what extent is this applicable?]
3. Cross-lingual training: parallel corpus used to train combined embeddings. Similar words will be mapped close to each other in same embedding space.
4. Joint optimization: Optimizing a combination of cross-lingual and monolingual losses.

For parallel data, we can have different types of data:

1. Word-aligned data
2. Sentence-aligned data
3. Document aligned data

4. Lexicon (translations between multiple language-tables)
5. No parallel data (only monolingual resources)

These are the ways to train word-embeddings (amongst others):

- word2vec variants
- skip-gram with negative sampling (SGNS)
- continuous bag-of-words (CBOW)
- GloVe matrix factorization approach

Learning a linear projection from monolingual embeddings and parallel data.

Projection via CCA (Canonical component analysis). A transformative matrix is learned for each language, the mapped space is the same space across different languages. Using 80% of projection vectors and using top k components generally yield highest performance.

Normalisation and orthogonal transport. Product similarity measure (correlation) can be seen as Wasserstein distance (because it measures how much data points are away from each other) $c^T c$. Vectors are normalised during training with unit vector. Because unit length, we can solve this by an orthogonal matrix. (Does any information go away? Maybe it's good to regualrize even further so that we know that the mapping is indeed possible?)

Max-margin intrudercs. Linear least squares leads to the hubness problem. Words tend to appear as nearest neighbors of many other words (-¿ perhaps some non-random sampling is best here? to move away). Use a ranking loss (sounds like optimal transport again...) Ranking loss is also used in wasserstein context.

Alignment based projection: Count number of times each word in source language is aligned with each word in target language document / sentence.

Adversarial autoencoder Autoencoder learns to re-create the embedding, while discriminator learns to maximize the projection

Orthogonal transformation, normalisation, mean centering Relation between the constraints is not clear. Starting with basic optimization objective, they add regualizers and loss functions as intuitively make sense. -¿ Dot-products need to be maintained after the mapping (as much as possible) constraint (i dont like) -¿ equal contribution to objective done by normaliizing the embedding.. -¿ two randomly selected words should in general not be similar

Pseudo-Cross-Label Captures interactions between words in different languages. Mostly skipped because brain capacity...

Joing training with 2 x 2000 dimensional input (one for each job-system)... can then translate each one in one latent space to another. How do we find parallel logic here, however..?

(-¿ Perhaps it makes sense to draw all the literature in the context of wassetstein, to do a probabilistic and principled analysis from there on.)

Observation (FRAGE: Frequency-Agnostic Word Representation): It is a feature that most-frequent words are embedded to different locations than non-most-embedded features. Xing et al argue there is a mismatch between the way it is learned, and the projection and loss-function. Tries to find a uniform theory across the three components.

If we use monolingual resources only, the common approach is

I focus on work which solely includes as input the bilingual word-embeddings. Because I want to devise an algorithm which can map from any "matrix column" to any other "matrix column" (i.e. token to token, or item to item), I will not include papers which take as input NLP sentences / corpora used to build the embeddings.

This work deals with some summary on how unsupervised bilingual lexicon matching was achieved in past papers.

I will give a short summary of papers sorted by year of appearance, and what additional contributions and observation were added since the last iteration.

Some terminiology before we start

online data implies that some sort of parallel corpora is available

offline data implies that we use pre-trained monolingual embeddings to arrive at a toen-translation task

The general goal of bilingual lexicon matching is to learn a shared embedding space where words possessing simiilar meanings are projected to nearby points.

Most existing methods focus on minimizing the distance between the two token-datasets using some (variation) of Wassterstein, Jensen-Shannon divergence, etc.

What are datasets that we can use for benchmarking

- MUSE dataset (as used in Zhou 2019 et al
-

Instead of mapping one system into another, we can also allow for joint-training of the word-embeddings.

(This resource is great!!! https://ruder.io/cross-lingual-embeddings/)

5.1 Discrete Methods

5.1.1 Loss in Translation

Finding a non-orthogonal mapping is more effective, especially with distance dictionaries

Orthogonality preserves distances between word-pairs.

Euclidean closest nearest neighbor is taken as the translation pair.

Hubs are words that appear too frequently close to other words. This retrieval criterion suffers from the hubness problem.

Removing the loss creates a discrepancy between inference and training.

(-¿ Why should a loss-function be symmetric, not be symmetric?)

(-¿ especially with chinese. perhaps the non-orthogonality captures the ambigious tokens..?)

From the original problem of finding an orthogonal projection, we assimilate the auxiliary problem of

the CSLS criterion looks as follows:

$$CSLS(x,y) = -2\cos(x,y) + \frac{1}{k}\sum_{\mathbf{y}'\in\mathcal{N}_Y(\mathbf{x})}\cos(\mathbf{x},\mathbf{y}') + \frac{1}{k}\sum_{\mathbf{x}'\in\mathcal{N}_X(\mathbf{y})}\cos(\mathbf{x}',\mathbf{y}) \tag{5.1}$$

We then use the objective function of a relaxed

$$\min_{\mathbf{W}\in\mathcal{O}_d}\frac{1}{n}\sum_{i=1}^{n} -2\mathbf{x}_i^\top\mathbf{W}^\top\mathbf{y}_i \tag{5.2}$$

$$+\frac{1}{k}\sum_{\mathbf{y}_j\in\mathcal{N}_Y(\mathbf{W}\mathbf{x}_i)}\mathbf{x}_i^\top\mathbf{W}^\top\mathbf{y}_j \tag{5.3}$$

$$+\frac{1}{k}\sum_{\mathbf{W}\mathbf{x}_j\in\mathcal{N}_X(\mathbf{y}_i)}\mathbf{x}_j^\top\mathbf{W}^\top\mathbf{y}_i \tag{5.4}$$

The optimization is over a convex manifold.

5.1.2 MUSE Facebook using a Min-Max objective

CSLS used in the training objective directly instead of using it just during retrieval.

Mikolov et. al learn a mapping between the two sets of Y and X:

$$W^\star = \text{argmin}_{W\in M_d(\mathbf{R})}\|WX - Y\|_F \tag{5.5}$$

Translation of any source token s to a target token t is defined as

$$t = \text{argmax}_t \cos(W x_s, y_t)$$

(i.e. we just create a mapping between x and t using a linear interpolation matrix and take the maximal scalar product.

Mikolov et al do not observe improved loss when using more advanced neural networks..

Enforcing an orthogonality constraint, this boils down to the procrustes problem as given by

$$W^\star = \text{argmin}_{W \in O_d(\mathbf{R})} \|WX - Y\|_F = UV^T \tag{5.6}$$

where U and V^T respectively the left and right Eigenvectors of YX^T.

We can create an adversarial unsupervised approach by taking random samples from

$X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$

where the discriminator has to detect which of the two datasets the sample stemmed from. The "generator" (not really a generator) W is trained to prevent the discriminator from making accurate decisions. Specifically, this is modeled as a two-player game, where WX and Y should be as similar as possible.

As such, the discriminator objective is:

$$\mathcal{L}_D(\theta_D|W) = \tag{5.7}$$

$$= -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(\text{source} = 1|Wx) \tag{5.8}$$

$$-\frac{1}{m}\sum_{i=1}^{n} \log P_{\theta_D}(\text{source} = 0|Wx) \tag{5.9}$$

and the generator objective is:

$$\mathcal{L}_W(W|\theta_D) = \tag{5.10}$$

$$= -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(\text{ source } = 0|Wx_i) \tag{5.11}$$

$$-\frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}(\text{ source } = 1|y_i) \tag{5.12}$$

Here, the discriminator is trained on the labels whether the word came from the first embedding, or the second embedding.

(-¿ Is there a mathematical way to simply this..?)

The resulting rotation matrix is not on-par with supervised approaches. As such, the authors propose to make use of the frequencies as additional information. As such, we use the most common words as anchors.

We then generate a dictionary from the two embeddings. Then we solve the procrustres to arrive at the final orthogonal matrix.

Use of the CSLS

5.2 Methods using Normalising Flows

5.2.1 Density Matching for Bilingual Word Embeddings (Zhou 2019)

Zhou et. al consider the problem by creating two smooth embedding spaces.

For simplicity, I will refer to embedding space $\mathcal{X} \in \mathbf{R}^d$ as the source embeddings, and the embedding space $\mathcal{Y} \in \mathbf{R}^d$ as the target embeddings.

We denote x_i and y_j as the individual words with respective embeddings $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

The general approach of the authors is as follows:

We learn two mapping functions $f_{xy} : \mathcal{X} \to \mathcal{Y}$, $f_{yx} : \mathcal{Y} \to \mathcal{X}$

We maximize the density of data points in the source space. We use a log-likelihood term in the loss-function.

For monolingual embeddings, we define tractable density functions $p(x)$, $p(y)$.

We use gaussian embeddings, where each single gaussian module represents a single word.

$$p(x) \approx p(y) \approx GMM \tag{5.13}$$

This implies that for both $\forall x \in \mathcal{X}$ and $\forall y \in \mathcal{Y}$ we have

$$p(x) = \sum_{i \in 1,\dots,N_x} \pi(x_i)\tilde{p}(x|x_i) \tag{5.14}$$

where the final probability term

$$\tilde{p}(x|x_i) = \mathcal{N}(x|x_i, \sigma_x^2 I)$$

describes the probability density of the single word-embedding, represented through a Gaussian in this case, and N_x defines the total number of components / words in dictionary $\mathcal{C}$.

To initiate the embeddings, the authors use the word2vec (WAS IT WORD2VEC) point-vector as the mean of the Gaussian distribution, and use a fixed variance term σ^2 for all Gaussians.

Furthermore, instead of using a uniform prior over the mixture weights $\pi(x_i)$, the authors use the observation that frequencies may give bettern information, as as such initiate the priors as the relative frequencies of the words, i.e.

$$\pi(x_i) = \frac{\text{freq}(x_i)}{\sum_j \text{freq}(x_j)}$$

.

This implies that the experiment is not executed in a manner where the Gaussian Mixutres are carefully chose, but rather by augmenting and generalizing the word2vec point vectors to Gaussian distributions. This has the effect that the resulting probability space becomes smooth (rather than a discrete space of point-vectors) [CITE??? SMOOTHNESS] Another assumption made is that a gaussian distribution is best-able to capture this mixtures.

Every training step obtains samples form the Gaussian mixture space.

The density calculation can be defined by using volume-preserving invertible transformations. Perhaps it could make sense to also apply convolutional operators?

To stabilize the unsupervised training, the author employs the following methods.

1. back-translation-loss to train both directions (this is conceptually similar to the cyclic dependency in GANs.
2. the author uses a weakly loss. identical text-strings in both languages are mapped to similar places in the space.

3. frequency of the words is used as a prior to the GMM prior weights.

The authors indicate robust training which is not strongly dependent on initialization

---

The training functions in such as a way as to increase the log-probability density. The general approach is:

1. A continuous vector x is sampled from the GMM. Specifically,

$$x_i \sim \pi(x_i), x \sim \tilde{p}(x|x_i) \tag{5.15}$$

2. We then apply f_{xy} to get the inferred target embedding $y = f_{xy}(x)$. Specifically, we define the mapping operation as:

$$f_{xy}(\cdot) = W_{xy}\cdot \tag{5.16}$$

   We choose W as an invertible matrix, thus, the inverse function is defined as

$$x = f(y) = W_{xy}^{-1}y \tag{5.17}$$

   setting the volume of this operation to

$$J(f^{-1}(x)) = W_{xy}$$

   which is used for the log-density calculations. Specifically

$$\log p(x; W_{xy}) = \log p(y) + \log|\det(W_{xy})| \tag{5.18}$$

3. We then maximize the log-likelihood, which equivalent to minimizing the expectation of the KL-divergence between the prior and modal distributions.

$$KL(p(x)||p(x; W_{xy}) \tag{5.19}$$

   We arrive at the conditional unsupervised mapping-loss function of

(incorporating the observation that each mixture represents a single word token)

$$\mathcal{L}_{xy} = \mathbf{E}x \sim p(x) \left[\log p(y) + \log \left|\det(W_{xy})\right|\right] \tag{5.20}$$

Where the first term is the log-density of the data, and the second term describes a regualizer (prior) over the sample model which we use for the mapping. This equation corresponds to the normalising flow which is volume-preserving.

The maximum log-likelihood is proportional to the minimum expectation of the KL-divergence between the prior and modal distribution. (DAFUQ IS THIS MODAL DISTRIBUTION)

5.3 Methods viewing this problem as an Optimal Transport (OT) Problem

5.3.1 Gromov-Wasserstein Alignment of Word Embedding Spaces

The problem of aligning word embedding spaces can also be seen as an Optimal Transport (***OT***) problem.

Some work in using the Gromov-Wasserstein distance function has been proposed (CITE Jaakkola).

The authors argue that the supervised case is

Algorithm 1: How to write algorithms

Result: Write here the result

initialization;

while *While condition* **do**

    instructions;

    if *condition* **then**

        instructions1;

        instructions2;

    else

        instructions3;

    end

end

Supervised Case: Procrustes

Assuming we have

$$T : \mathcal{X} \to \mathcal{Y}$$

such that

$$T(x^(i)) \approx y^j$$

where w_j^y is a translation of w_i^x.

Let $\mathbf{X}$ and $\mathbf{Y}$ be the matrices with the column-vectors describing the word-embeddings.

We can then find T by solving the equation

$$\min_{T \in \mathcal{F}} \|X - T(Y)\|_F^2 \tag{5.21}$$

THe quality of the resulting alignment depends on how well we can find T, as well as on the choice of the space $\mathcal{F}$.

T orthogonal matrix is a popoular choice, results iin the orthogonal Procrustes problem

$$\min_{P \in O(n)} \|X - PY\|_F^2 \tag{5.22}$$

with $O(n) = P \in \mathbf{R}^{nxn} | P^T P = I$.

In this case, we can find a closed-form solution in terms of the SVD of PX and Y. Given an SVD of $U\Sigma V^T$ of XY^T, the solution is given by $P^* = UV^T$.

Because the loss must be able to be measured, it only solves the supervised problem.

Unsupervised Maps: Optimal Transport

Optimal transport formalizes the problem of finding a minimum cost mapping between two point sets, viewed as discrete distributions.

We assume two empirical delta-distributions over embeddings (which are as such equivalent to point-vectors).

WHAT WAS THIS SYMBOL AGAIN WHICH LOOKS LIKE A V

$$\mu = \sum_{i=1}^{n} p_i \delta_{x^(i)}, v = \sum_{j=1}^{m} q_j \delta_{y^(i)} \tag{5.23}$$

where p_i and q_j can be considered as the prior distributions, usually set to the uniform distribution or word-frequency.

The transport problem is then realized as

$$\inf_T \{c(x, T(x)) d\mu(x) | T\mu = v\} \tag{5.24}$$

where the cost is usually simply defined as $\|x - T(x)\|$ and $T\mu = v$ implies that the source points must exactly map to the target points.

We use Kantorovich's relaxed formulation, where the set of transportation plans is a polytope:

$$\prod(p,q) = \{\Gamma \in \mathbf{R}_{+}^{n\times m} | \Gamma \mathbb{1}_n = p, \Gamma^T \mathbb{1}_n = q\} \tag{5.25}$$

In the end, this is just a matrix-factorization problem of the type, including a regularization on the total entropy H of the loss:

$$\min_{\Gamma\in\prod(p,q)} \langle \Gamma, C\rangle - \lambda H(\gamma) \tag{5.26}$$

The authors propose "Transport across unaligned spaces", addressing the need for vectors across two spaces. Instead of samples between two spaces, we now focus on the distance of the two spaces as a whole.

Upsides:

1. Discrete Vectors

2. Some formulation of "minimal" rotation

Shortcoming:

1. An overall single rotation will most likely not solve this issue

How do we now proceed with using the distances between two functions as the main information to map the two words.

5.3.2 Graph translation using normalizing flows

This follows from the observation that any document x token matrix can be modelled as a graph.

5.3.3 Using monolingual word-embeddings to map one to another

Cross-lingual word-mappings (Artetxe 2018)

Extension of work for more distant language pairs for looking at tokens.

Artetxe does the following series of oprations

1. Embedding normalization: Normalizes the length of the word-embeddings and then mean-center each dimension.

2. Fully unsupervised initialization: Uses distances to measure similarity ($M_X = X^T X$ and $M_Y = Y^T Y$). Common alternatives include finding common letters and numerals across two words. Distances should be isometric up to some permutation. Main assumption: Distance between the two spaces must be equivalent, else unsupervised is not possible! Compute sorted($\sqrt{M_X}$, $\sqrt{M_Y}$). This didnt quite work, so we randomly keep some elements with probability p. This allows for easier matching because there are many ways to generate this.. [This is basically like batch-sampling for training, no? This noise (if batchsize is chosen well), should be able to well-solve this]. We only focus on the top 20000 words in each language for simplicity. Instead of kNN, we use the CLSL retrieval. Use bidirectional learning to avoid local optima where one word is not available in another language.

3. Re-weighting [What??] by whitening the cross-correlation matrix. Ultimately, this again also just performs on distances.

$$USV^T = X^T DZ$$

looking at $W_X = US^{\frac{1}{2}}$, then we whiten by

$$U^T (X^T X)^{\frac{1}{2}} U$$

.

CSLS often used because of the "hubness" problem [Perhaps we can copy all these oprations in a probabilistic context with Wassterstein?]

Bilingual Lexicon Induction through Unsupervised Machine Translation (Artetxe 2019)

Uses machine translation to build a lexicon induction model.

Input to algorithm:

1. Monolingual corpora fasttext through vecmap and the corpora used to train.
2. 400'000 most common bigrams and 400'000 most common tri-grams.
3. To find most similar pairs, instead of using kNN CLSL, proposed is

$$\phi(\bar{f}|\bar{e}) = \frac{\exp(\cos(\bar{e}, \bar{f})/\tau)}{\sum_{\bar{f}'} \exp\left(\cos\left(\bar{e}, \bar{f}'\right)/\tau\right)}$$

   to extract the top 100 sentences 5-gram language model.
4. The proposed method increases this by at least 10 percent points (P@1).
5. underlying language model uses 2000 sentences from each monolingual corpora, a language modeling loss, and combines this with a cyclic consistency loss.

5.4 Gaussian Embeddings and Token matching in other applications

5.4.1 Creating Gaussian Embeddings to represent Graphs

(DEEP GAUSSIAN EMBEDDING OF GRAPHS: UNSUPERVISED INDUCTIVE LEARNING VIA RANKING)

Chapter 6

Metric learning on BERT embedding

The question is, do we need to apply domain adaptation method for BERT embeddings.

I would say no as a first response, as the clustering methods seem to work well across dimensions (they dont fail).

However, they could be better. We will see if a learned distance measure will facilitate with this.

Chapter 7

Analysis of the current state of the art

For the problem statement to be - generalizable to other datasets - unsupervised (perhaps some [noisy] inference on starting points possible..)

Some challenges that are faced are

- Hubness problem (MUSE, and CSLS)

(https://ruder.io/cross-lingual-embeddings/)

Functional modeling may be difficult to model. It may be able to model the relationship between words well, but it will not be able to distinguish the context or actual usage of the word within a sentence.

Word-order is completely ignored. [will probably ignore this aspect, simply because we dont want sequence-based approaches for generalizability] This may lead to different types of training..

Compositionality: word-representations can not be generalized further to sentence-generation and document-generation. Simply adding word-embeddings cannot result in a meaningful

Polysemy: a word may have multiple meanings. in a cross-lingual embedding

space, this feeling is amplified. there's some work in multi-sense embedding. this should enable to capture more fine-grained embeddings embeddings.

Feasability: Maybe not possible?

Upadhyay et al. argue that the choice of data is more important than the actual algorithm.

Include an analysis of all loss functions.

Include a list of assumptions made.

Include a list of observations made

Ablation study.

Unnormalized embeddings can modify training in that the loss function puts more weights to the high-occuring vectors.

Maybe Mikolov 2013b did not do proper hyperparameter tuning to check if deeper layers improve this performance..?

Definitely also look into this, [Analyzing the Limitations of Cross-lingual Word Embedding Mappings] seems to be an analysis of the difficulties etc.

Although perhaps a bit risky to cite, this also seems to be critical of current method. Would really need to double-check sources [EMPIRICAL OBSERVATIONS ON THE INSTABILITY OF ALIGNING WORD VECTOR SPACES WITH GANS]

Not sure if optimal transport is the solution. The i.e. counterexample on a 2-d space, where "one boot" (italy topology) and another boot are rotated by 180 degrees. Then optimal transport would find a wrong solution (because minimal transport!)...

Wasserstein seems to work well, and any measure that incorporate intra-correlation (Artetxe), which can be seen as a generalization to wasserstein. However, optimal transport I'm a bit skeptic of. Especially for distant measures it does not deliver nice results.

Chapter 8

Our Method

Ideas:

1. Generally, deviate more from first proposed paper.
2. (like EM) two-step training where dictionary is learned and then the underlying soft probability distribution (using parametrized models)
3. Instead of cross-correlation, look for some distance in the probability space which makes sense? Test multiple ones, and possibly arrive at a prior which is interesting
4. order of phrasing is different in korean / turkish to german
5. Enforce some sort of probability density on top?
6. Go deeper perhaps?
7. Generally collect a set of constraints and observations that seem to work, then combine them in a principled approach
8. Normalising flows seem invertible. Can we not just use $f_{xy} = f_{yx}$.
9. AutoML on normalising flows..?
10. Leave out individual variables
11. Jointly train both embedding and normalising flow

12. After two possible ways of embeddings are learned, can we use the weights of the neural networ (i.e. word2vec") to find a mapping that goes from one space into another? (... yes, we should be able to using least squares projection...)
13. Work on generalizability of this on other datasets (such as ESCO to australia, or Amazon Austria to Amazon Germany)
14. Regualizer that adds up to respect some corpus / frequency matrix properties...
15. Hubness problem by enforcing a prior to the dataset s.t. the embeddings are uniformly distributed in space?
16. Probability density matching using optimal transport not on Euclidean space, but on a Riemann manifold / hyperbolic space?

Other lines of work

1. Create thesaurus from language model
2. for each word, create a sentence in the thesaurus through the language model
3. iterate through a bunch of sentences. Make mean-variance from context-vectors to get uncertainty, and capture polysemy. dude, this is so smooth. -¿ i.e. we get word-embeddings from multiple documents.
4. A

After first meeting ideas:

-¿ Perhaps it makes sense to put a prior on BERT or ELMo for distill-learning to regualizre it to a certain space..?

Chapter 9

Further Work

Can be used for more unstructured data, like graphs.

Chapter 10

Evaluation

Chapter 11

Conclusion