LexSub:Word meaning in Context

Akanksha Bansal
The University of Texas at Austin
bansal@cs.utexas.edu

Gaurav Nanda
The University of Texas at Austin
nanda@utexas.edu

I. Abstract

In this paper, we present a comparative study of distributional similarity vs a graphical model based approach to generate lexical substitutions, i.e replacement of a target word in context with a suitable alternative substitute. The meaning of a word can be defined to be a posterior distribution over the paraphrases for the target word in context. We utilize the framework of probabilistic graphical models to define the meaning of context and conduct marginal inference. We use various distribution vector based factors representing the context and compare the divergence of the paraphrase vectors from the target in order to select the best candidate replacement. We then evaluated this model on the Lexical Substitution task 2007 (LexSub) data, which shows decent results. The biggest strength of the model is its ability to effectively capture the syntactic and semantic relationship between words.

Keywords—Paraphrases, probabilistic graphical models, probabilistic inference, loopy belief propagation.

II. Introduction

Word sense disambiguation (WSD) is the dominant task in the sub-field of computational linguistics that deals with the meaning of words in context.

In the recent work done by [2], [3] and [4] the meaning of the word *w* in context *c* is represented by a vector obtained by combining the vectors of *w*(word) and *c*(context) using some operation such as component-wise multiplication or addition.

We however try to take a step further, as has been done by [5] and build a model which draws inference on the meaning of the word by constructing a factor graph of the words in the sentence. For our project, we make use of distributional vector representations of the word to determine the factor function. These factors quantify the meaning of the *w* given the *c*. We aim to build a flexible platform that can be used to explore different relationships between the *w* and *c*, without depending on the method used to compute meaning of the words.

III. Problem Statement

Natural-language inference is highly context-sensitive, the applicability of inference rules depends on word sense and even finer grained contextual distinctions in usage [1]. While closely related to WSD, lexical substitution does not rely on explicitly defined sense inventories, the possible substitutions reflect all conceivable senses of the word, and the correct sense has to be ascertained to provide an accurate substitution.

These systems are given a naturally occurring sentence that contains a target word of interest and a disjoint candidate sense inventory, generally a list of dictionary definitions for the word. The goal of the system is then to choose the item out of the inventory that best fits the target word in the sentence. The system is evaluated based on some aggregate measure over the systems output on all the target words for all the sentences that it has been handed to disambiguate.

Work done in this field has shown that making use of dictionary meanings of word is not able to capture the complete meaning of the word. Therefore research is being done to create computational models representing word meaning, which do not represent a word instance through a single best sentence.

IV. Previous Work

Distributional models compute a single vector for a target word, which contains co-occurrence counts for all the occurrences of the target in a large corpus. We make use of polysemy-aware approaches that is using syntactic context similarity as has been described in [4]. Then in order to distinguish between multiple senses of the word in various parts of the speech, we define the vectors with regards to the POS tag of the words. *Ambiguity is an inherent component of human language and consequently these systems will at some point have to address this issue somehow.*

A. Task

English Lexical Substitution dataset (LexSub) is composed of a corpus labeled by asking multiple annotators to propose substitute paraphrases for a target word in a sentence. We evaluate our model on predicting suitable replacements for the sentences from this task. The task involves two subtasks:

1) Finding the *best* candidate given the context of the word token.
2) Finding the *set* of *ten* candidate substitutes for the word.

A diverse set of contexts are present in the dataset sentences . Therefore this task is a means of examining the issue of word sense representation by providing an unbiased platform to all, that evaluates the inventories and contextual disambiguation models.

Because of the diversity in the sentences, only systems which take into account the context to determine the best possible replacement word, would be able to perform better on this task. The most popular way to accomplish this task is by making use of a predefined inventory such as WordNet and then ranking using a Language based model to rank the substitutions.

We however make use of both distributional vectors and Big Huge Thesaurus in our graphical representation to get the best possible replacement words.

V. Graphical Model

We make use of a factor graph to represent a sentence. A factor graph is a bipartite graph over two types of nodes, nodes that correspond to variables (circles) and nodes that correspond to factors (filled squares). A factor is a function whose arguments are the variable nodes adjacent to the factor node. The factor graph as a whole represents the product of all the factors in it.

We represent each content word of the sentence through two adjacent nodes: the **observed node** represents the surface form of the word itself, and the **hidden node** represents its usage meaning. The distribution over values that we infer for the hidden node is a paraphrase distribution for the observed word.

The model derives a separate representation for each word in context, rather than a joint representation for a phrase. Such models are referred to as word usage models. Here each word representation is context dependent and inferred dynamically.

The distribution inferred over the hidden node given the evidence from its observed context, the paraphrase distribution of the observed word is determined. The observed context can be anything and may include evidence from the words heads and dependencies, its left and right context, or the entire document. *For a fair comparison to our previous work, we just make use of the left and right context words with a window size of 2.* The strength of the model lies in the ability to add any kind of knowledge to the model by using various factors. For example, we can have factors represent the subject-object relationship between the words, or we can have topic variable. These values can be integrated into the model by summing over all the hidden variables and multiplying the corresponding factors.

Given below is an example of how the graph is generated for a given sentence: ”THE BOX WAS IN THE PEN”.

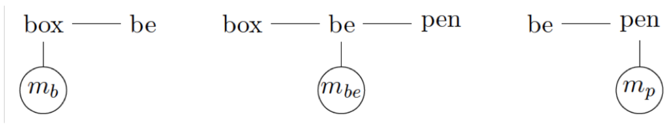

Fig. 1. A dependency parse of sentence

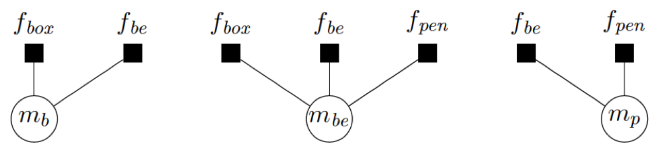

Fig. 2. A Factor Graph of the dependency parse of sentence

Figure 1 represents a dependency parse for the sentence after lemmatization and trimming of function words. Figure 2 shows the factor graph to which we map the dependency graph. As mentioned before, each content word from the sentence is associated with a paraphrase node, a hidden variable whose values can range over the whole vocabulary. For example, the paraphrase node associated with ”box” is m_b. Each content word is additionally associated with a unary factor, the word factor, representing the observed word. For box this is f_{box}. This factor is linked to m_{be} to model the influence of the observed word on the paraphrase distribution. The values of m_b are restricted to the set of valid paraphrases for w_b(box), and which are derived from external knowledge sources.

Now lets take into account the sentence : ”THE PLAYER RAN TO THE BALL”.

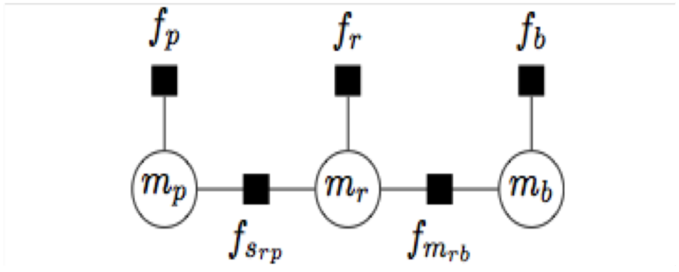

Fig. 3. Factor graph of sentence making use of binary factors

Figure 3 introduces binary factors into the graph. These factors are called **selection factor**. For example, the f_{srp} edge is a binary factor linking m_r and m_p. This way the factors are able to capture the relation between the paraphrase nodes. This graph is mostly beneficial in capturing the effects of paraphrase nodes which are not directly connected. Eg.

1) The player ran to the ball.
2) The debutante ran to the ball

Here the word *player* and *debutante* affect the meaning of the word *ball*, which can only be captured by a binary factor representation. It models the influence of *selectional preferences* on mutually contextualizing *player* and *ran*. These are able to determine to what extent each value of m_p is compatible with each value of m_r. Overall, for the random variables m_p;m_r;m_b the graph has the following factorization of the global function F:

$$\begin{aligned} F(m_p, m_r, m_b) = f_{w_p}(f_p) * f_{w_r}(f_r) * f_{w_b}(f_b) \\ *f_{s_{pr}}(m_p, m_r) * f_{s_{rb}}(m_r, m_b) \end{aligned}$$

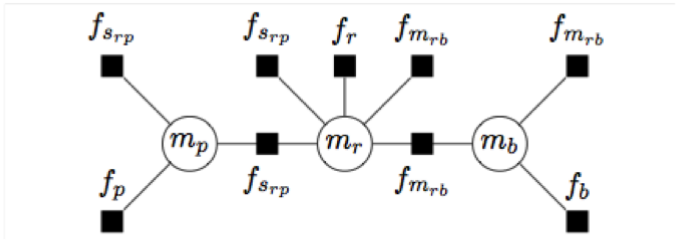

Fig. 4. A combined Factor graph of the sentence

Figure 4 represents a combination of both the models represented above. It is a combination of the paraphrase transformation and the adjacency transformation. It models both co-locational strength of adjacent observations (i.e. at) as well as generalized information from the entire sentence (pt).

Dependency graph results in the following factorization of the global function:

$$P_G(m) \propto (\prod_{(i,j)\in E_G} f_{r_{ij}}(m_i, m_j)) \prod_{i \in V_G} (f_{w_i}(m_i) \prod_{j \in \Gamma_i^h} f_{r_{ji}}(m_i) \prod_{k \in \Gamma_i^d} f_{r_{ik}}(m_i))$$

where Γ_i^h denotes the set of heads for node i in the original dependency parse and Γ_i^h denotes the set of dependents.

A. Marginal inference

Given the set of random variables m and a measure of its information $F_s(m)$ in relation to another set of random variables s, we need to determine a specific random variable $m_i \in m$ solely in relation to s. This can be obtained by marginalizing out (i.e. summing over) the random variables that are not m_i: $F_s(mi) = \sum_{m/\{m\}} F_s(m)$

The left hand side of this equation is the marginal of m_i and such functions as a whole in relation to $F_s(m)$ (called the *global function*) are referred to as marginals. This is an intractable problem, but if some random variables are independent of each other or are assumed to be independent, the ordering of the summation can be carefully arranged so as to make this pliable.

Graphical representations of these independence assumptions in factor graph helps formalize and visualize the models that derive from them.

In graphs that are trees or poly-trees, the sum-product algorithm can be used for inference. However, in our case, dependency parsers generated graphs that are not poly-trees, therefore the graph may contain loops. The [5] make use of loopy belief propagation to approximate marginals. Then they make use of Jensen-Shannon Divergence to measure the divergence between distributions determined in between the original sentence and the one inferred for the paraphrase nodes.

Taking the previous example further, the paraphrases possible for *ball* are : *globe*, *dance*, etc. Then we run inference to determine the distribution possible for sentences:

1) The debutante ran towards the globe.
2) The debutante ran towards the dance.

We conduct inference on each of the new sentences and compare the distribution inferred over *ball* with each of the paraphrases in terms of Jensen-Shannon divergence (JS), since the values of JS are non-negative and the function is symmetric over its arguments.

We define s_{dance} to be the sentence with $dance$ in place of *ball* and s_{ball} to be the original sentence. Then we define $P(m_{dance}|s_{dance})$ and $P(m_{ball}|s_{ball})$ to be the probability mass functions inferred for the paraphrase nodes of ball and dance, respectively, given the observed sentences. Then we measure JS between the two distributions. We also measure distances between $ball$ and $dance$, $ball$ and $globe$ and so forth for all possible paraphrases m_{ball}.

JS, being a measure of difference, will place larger values on more dissimilar distributions whereas we want larger values on more similar distributions.

B. Context Factor Generation

We make use of prototype-based approach where we compute a (prototype) vector for each word in isolation using the co-occurrence matrix, then modify this vector according to the context in a given sentence [10].

Once we have defined a function to compute paraphrase factor for a given word, we create similar factors using all the candidate substitutes. The next sections cover how exactly the paraphrase factors are calculated.

1) Paraphrases Generation: The first step is to collect set of potential replacement candidates for a given target word using the Big Huge Thesaurus [13]. The replacement words for the context words are also collected. All the possible paraphrases are selected and further used in factor's value computation in the later steps.

2) Paraphrase Vector Representation: We create a distributional vector model to represent word meanings. Each lemma is described as a high dimensional vector that records co-occurrence of context features over a large corpus. The pipeline from corpora to semantic space can be roughly split into two major steps: pre-processing the corpus to collect the relevant counts, and processing the extracted counts mathematically.

3) Data-set : We need a large data-set to construct co-occurrence matrix used in generating distributional similarity model. We used two different datasets, Wikicorpus by [6] and Uwack by [7] to create the distributional space. Wikicorpus contains approximately 750 million words corpus (around 10.GB of data) and Uwack is five times the size of WikiCorpus. Using two different corpora enabled us to integrate a diversity in the distributional space.

4) Pre-processing: To leverage the syntactic relation of POS tagged words, we suffix all the words with first letter of the Penn Treebank codes for various POS tags. This way, we are able to capture the polysemic occurrences of single word and categorize them based of the the POS category. We only include Nouns, Verbs, Adverbs and Adjectives in our vector space as we believe that these have highest impact on the words in context. Thus we do filtering which is much more restricted than just Function word removal. As our data source is based on Wikipedia, we also pre-process our data to exclude all the non-dictionary words.

5) Semantic Space Generation: After creating co-occurrence files, we use python based DISSECT [9] toolkit to construct the semantic space. Then, we apply Positive PointWise Mutual Information (PPMI) weighting scheme for feature selection.

$$pmi(r, c) = \log \frac{P(r, c)}{P(r)P(c)} \tag{1}$$

$$ppmi(r, c) = pmi(r, c) if pmi(r, c) \geq 0 else 0 \tag{2}$$

[8] demonstrated that PPMI performs better than a wide variety of other weighting approaches when measuring semantic similarity with word context matrices.

C. Inference

The sum-product and max-product algorithms give exact answers for tree graphical models, but if we apply the same update rules on a general graph, it often gives pretty reasonable results. This is known as loopy belief propagation. The initialization and scheduling of message updates is adjusted slightly because graphs might not contain any leaves.

Because it is not possible to perform exact inference of the marginal for m_i given a transformed dependency parse with loops, Loopy Belief Propagation instead approximates the marginal of m_i at some iteration $t+1$ based only on the values of the approximate marginals of its neighbors from the previous iteration t. We use OpenGm2 [12] to represent our factor graph and perform the approximate marginalization.

$$P^{(t+1)}(m_i) = C(m_i) \prod_{j \in \Gamma_i^h} \Big(\sum_{m_j} f_{r_{ji}}(m_j, m_i) P^{(t)}(m_j)\Big) \prod_{k \in \Gamma_i^d} \Big(\sum_{m_k} f_{r_{ik}}(m_k, m_i) P^{(t)}(m_k)\Big)$$

$$C(m_i) = f_D(m_i) f_{w_i}(m_i) \prod_{j \in \Gamma_i^h} f_{w_j, r_{ji}}(m_i) \prod_{k \in \Gamma_i^d} f_{w_k, r_{ik}}(m_i) \tag{3}$$

VI. Results

In order to evaluate the performance of the model for the task, we submit for each of the sentence in the test set (1700 sentences) a list of words which can replace the target word. The replacements are assumed to be ranked in the order of their likelihood of occurrence, as the replacement word. The better the set of candidates, the higher the chance that one or more appropriate replacements are found for the given context.

We measure the precision and the recall for four subtasks: *best normal*, which measures the precision and recall obtained when the first replacement provided by the system is selected; *best mode*, which is similar to *best normal*, but it gives credit only if the first replacement returned by the system matches the replacement in the gold standard data set that was most frequently selected by the annotators; *oot normal*, which is similar to best normal, but it measures the precision and recall for the top ten replacements suggested by the system; and *oot mode*, which is similar to best mode, but it again considers the top ten synonyms returned by the system rather than just one. For oot, our system does not report duplicates in the list of best ten candidates. If we do report duplicates in the results, we can artificially inflate the precision values by reporting best substitution ten times. [11] suggest that, by reporting duplicates, we can theoretically achieve precision of 457% .

System	best normal	best mode	oot normal	oot mode
LRAdd	**15.61**	**26.67**	**40.8**	**57.56**
Multiplication	8.23	13.79	24.46	35.96
Addition	12.78	22.28	38.31	53.01
Graphical Model	**8.30**	**14.85**	**29.91**	**42.60**

TABLE I. COMPARISON OF VARIOUS CONTEXT VECTOR GENERATION METHODS.

Table 1 contains the comparison of our graphical model against the three different models that we had tried earlier. *Multiplication* and *Addition* models would multiply/add the respective context vectors to find the best candidates in the distributional space. *LRAdd* model which is a combination of both multiplication and addition models, produced *state-of-the-art* results.

A. Discussion

Our graphical model produces decent results but not as good as LRAdd model. Specially, the results for BEST metric are really not that great. On careful observation, we see that graphical model while ranking the candidates gets biased towards the higher value of unary similarity factors. Therefore, we tend to loose the contextual effect. However, if instead of using just the left and right context vectors, we use dependency graphs as discussed in the [5], we expect the performance to improve. Though that should improve performance for all the above mentioned models.

We also observed that the values are not as good for the verbs when compared to nouns, adverbs and adjectives. This could be attributed tot he fact that verbs tend to have a much more wider distribution when compared with other parts of speech. Thus making the inference more difficult and not as accurate.

You can refer to our code in the github [14].

References

[1] Szpektor, Idan and Shnarch, Eyal and Dagan, Ido Instance-based evaluation of entailment rule acquisition, 2007

[2] Thater, Stefan and Fürstenau, Hagen and Pinkal, Manfred, Contextualizing semantic representations using syntactically enriched vector models,2010.

[3] Mitchell, Jeff and Lapata, Mirella, Vector-based Models of Semantic Composition, 2008.

[4] Erk, Katrin and Padó, Sebastian, A structured vector space model for word meaning in context, 2008.

[5] Moon, Taesun, and Katrin Erk. An inference-based model of word meaning in context as a paraphrase distribution. ACM Transactions on Intelligent Systems and Technology (TIST) 4.3 (2013).

[6] Reese, Samuel and Boleda, Gemma and Cuadros, Montse and Padró, Lluís and Rigau, German, Wikicorpus: A word-sense disambiguated multilingual wikipedia corpus, 2010.

[7] Baroni, Marco and Bernardini, Silvia and Ferraresi, Adriano and Zanchetta, Eros, The WaCky wide web: a collection of very large linguistically processed web-crawled corpora, 2009.

[8] Bullinaria, John A and Levy, Joseph P, Extracting semantic representations from word co-occurrence statistics: A computational study, 2007.

[9] Dinu, Georgiana, Nghia The Pham, and Marco Baroni. 2013a. DISSECT: DIStributional SEmantics Composition Toolkit, 2013.

[10] Thater, Stefan and Dinu, Georgiana and Pinkal, Manfred, Ranking paraphrases in context, 2009.

[11] McCarthy, Diana and Navigli, Roberto, The English lexical substitution task,2009.

[12] OpenGm2, http://hci.iwr.uni-heidelberg.de/opengm2/

[13] Big Huge Thesaurus, https://words.bighugelabs.com/

[14] https://github.com/gaurav324/English-Lexicalized-Text-Substituion/