Master's Degree programme
in Computer Science

Final Thesis

A Game Theoretic Approach to Multimodal Verb Sense Disambiguation

Supervisor
Ch. Prof. Marcello Pelillo

Assistant Supervisor
Ch. Prof. Sebastiano Vascon

Graduand
Gianluca Bigaglia
Matricolation number
854406

Academic Year
2018/2019

Abstract

Verb Sense Disambiguation is a well-known task in NLP, the aim is to find the correct sense of a verb in a sentence. Recently, this problem has been extended in a multimodal scenario by exploiting both textual and visual features of ambiguous verbs. The sense of a verb is assigned by the actual content of the image paired with it. In this work, such a task will be performed in a transductive semi-supervised learning (SSL) setting in which a small amount of labelled information is used to perform the verb-sense classification. The SSL is performed through a game-theoretic framework, in which each multimodal representation of a pair image-verb is a player and the possible strategies correspond to the set of senses that the verb belongs to. A Nash Equilibrium in this non-cooperative game corresponds to a consistent labeling between verbs and their possible senses. Experiments have been carried out on the recently published dataset VerSe. The results achieved outperform the current state-of-the-art by a large margin.

Contents

Chapter 1

Introduction

1.1 Word Sense Disambiguation (WSD)

Each language has ambiguous words (i.e. words with multiple meanings), the human brain has a natural ability to identify the meant sense of words in a sentence. This process is unconsciously carried out using the words that occur close to the ambiguous word (i.e. word context). For instance in these sentences:

1. The tap is dripping, there is a *break* in the pipe system.
2. When we were halfway to Milan, the driver decided to have a *break*.

the word *break* occurs in both sentences, however, it is utilised in two different ways: In the first case, it is employed to denote a hole in a pipe, whereas, in the second case it is used as a synonym for *pause*. It is clear that in both occurrences the role of the word is to indicate an interruption on something, nevertheless, such usages are different.

In the field of *Natural Language Processing* (NLP) i.e. the Machine Learning area which deals with the interpretation and understanding of human communication; there is subsphere of research which aims to study *Word Sense Disambiguation* (WSD). WSD is the task to identify the proper meaning of one or more words in a sentence in a computational manner among a batch of senses; this is done by exploiting the context in which such words occur.
This task is needed to get unstructured and semi-structured textual data (data warehouses, web pages, document corpora) machine-readable. The need of WSD systems arose when the data on the web increased so much that typical text mining techniques were not sufficient to handle and understand such a wide amount of information, so new methods were needed [49]. For this reason, WSD cannot be considered as a standalone task, in fact, it acts as a preliminary process for some other applications which rely on it, e.g. *Machine translation [52], Information Extraction [47], Content Analysis, Sentiment Analysis and Speech Synthesis [48].*
WSD problem has been considered for the first time in the half of '900s by Warren

Weaver for a machine translation application [46]. At that time, researchers started taking into account aspects like word contexts and patterns. Nevertheless, they had problems as limited computational power and the lack of wide and structured knowledge sources [49]. The decisive moment occurred in the '80s, with the introduction of large knowledge bases such as dictionaries, thesauri, and labelled corpora [48].

The first attempts to tackle the WSD problem was symbolic rather than data-driven: semantic networks were used to represent sentences. Nevertheless, this class of techniques requires the manual encoding of rules and concepts [48].
There are several approaches which can be categorised depending on the amount of knowledge needed at either end of the scale are domain-driven methods (symbolic) and unsupervised methods. However, there are three common WSD approaches [50]:

- Supervised
- Unsupervised
- Knowledge-based

1.1.1 Supervised word sense disambiguation

These methods rely on sense-tagged corpora which act as the training set. The goal is to train a classifier being able to tag words with their correct sense in an unlabeled corpus. Supervised systems require *a priori* knowledge about all the available senses of target words. Therefore, each future prediction is based on senses (labels) covered in training data. These methods usually extract features like the target word context i.e. the collocation of words occurring near it, part-of-speech tags, to recognise the syntactic role of words in the sentence.

Being a classification task, in WSD can be exploited all typical Statistical Learning Theory-based techniques i.e. Naive Bayes, Max Entropy [51], Support Vector Machines [5], and other ones like Decision Lists [53] and Neural Networks [4, 77]. A typical very simple algorithm which can be used is the Most Frequent Sense (MFS) heuristic, which usually acts as a baseline to evaluate more complex models. It classifies the sense of the target word according to the sense which occurred the most in the training corpora. Clearly, with this technique, there is no need to take into account the context or other information beyond the verb. The results are usually better than WSD algorithms due to the skewness of word senses distributions. Nevertheless, this approach cannot generalise.

1.1.2 Unsupervised word sense disambiguation

The main limit of supervised WSD systems is that they have to handle all the possible senses of target corpus and it is unlikely to find a training set with a sufficiently big sample size for each word sense [49]. For this reason, supervised WSD operates better when only a part of the corpus is going to be disambiguated like either nouns or verbs. So, when the number of senses to handle become unfeasible, unsupervised learning algorithms may be a more suitable solution. The goal of this class of systems is to disambiguate word senses of an unlabeled corpus from which they are extracted. Since the labels are usually not known *a priori*, the unsupervised approach may handle better *all-world WSD* tasks[1].

Purely unsupervised systems rely on sense distributions of the target word in a specific corpus, exploiting the fact that words with the same sense will have similar neighbouring words. However, since they do not rely on any machine-readable resource, rather than labelling words with senses, they can only extract clusters of senses. So, once that word contexts are encoded into vectors and a distance metric is defined, standard clustering algorithms can be used. For instance, one of the most used clustering algorithms for WSD is agglomerative clustering.

The main limit of purely unsupervised WSD algorithms is that since they do not rely on any knowledge bases, the extracted senses are likely to not match the ones categorised and defined in standard dictionaries. In these situations, the WSD task is switched into a *Word Sense Discrimination* or *Word Sense Induction* (WSI) task [54] and the results can be difficult to compare with the ones of supervised methods.

1.1.3 Knowledge-based word sense disambiguation

This category of methods is the most attracting one in the research community. The amount of data available is the same as in unsupervised learning, but their senses are regulated using wide knowledge bases, resulting in a proper WSD. The main difference with respect to purely unsupervised algorithms is that rather than extracting the sense inventory from the corpus, it is known *a priori*. So, a mapping between a dictionary and the occurrences in the corpus is performed. The main categories of knowledge bases which can be used are [49]:

- **Machine Readable Dictionaries:** the digital version of a canonical dictionary. So every word is paired with a list of senses in which each of them has a definition and possibly some examples of use.
- **Thesauri:** they provide information about relationships between words, e.g. synonymy, antonymy, etc. In a practical view, each sense of a word has a list of semantically related words.
- **Ontologies:** lexical databases usually made for domain-specific terms. Here, concepts are connected through semantic networks and are organised according to a taxonomy.

These resources have a great performance impact on purely unsupervised learning algorithms. For instance, they can be combined with the input corpus to extract

[1]In all-world WSD, all the words that compose a sentence have to be disambiguated, whereas, a simpler instance of the WSD problem is to consider either nouns or verbs or some specific words.

the text domain and find which is the more frequent sense of the target word in that scenario [55]. Some other approaches compute which is the sense with more overlapping words between the target word context (neighbourhood) and the context of each sense entry in a sense inventory i.e. the Lesk algorithm [15] (a more detailed description can be found in Section 2.1.3). Knowledge bases can also be used to provide domain-free and corpus-independent sense distributions. For instance, in WSD performance evaluation, a canonical baseline method that relies on knowledge bases is the First Sense (FS) heuristic (sometimes it is called MFS anyway). Such method returns the first sense listed in the knowledge base, which should be the most common for the language.
Even if supervised methods generally perform better, knowledge-based methods [59] and graph-based methods [24,57,58,60] are gaining more and more interest and their results are getting closer to the supervised counterpart [56]. Such algorithms do not need training corpus, but they use the lexical knowledge and semantic relationships extracted from the knowledge-base and perform WSD by exploiting graph theory. The most important resources which are currently used in WSD for these systems are ontologies like WordNet, OntoNotes and BabelNet [9–11].

WordNet WordNet was born as a pure lexical database whose goal is to exploit conceptual-driven search instead of the canonical alphabetical-driven one. However, once its potential has been shown, it evolved into a dictionary based on psycholinguistic principles. WordNet is organised in categories, the first distinction is between parts of speech: *nouns, verbs, adjectives, adverbs, and function words*. Each one of these classes has a different organisation, for instance, nouns are classified in hierarchies, verbs are organised in entailment relations, and so on [10]. Moreover, each word entry has lists of semantic relationships (antonymy, synonymy, pertainymy, etc.).
WordNet can be considered as the evolution in terms of organisation of Machine-Readable Dictionaries, combined with the relationships of thesauri; becoming an essential resource in WSD.

OntoNotes Another big knowledge base is OntoNotes [9]. Even if it was meant for a different task, it became a WSD resource. The main difference with WordNet is that their fine-grained categories produce a low agreement between human annotators and automatic tagging systems. OntoNotes instead relies on WordNet categories and split them into more coarse-grained senses. This split improved the *inter-annotator agreement*.

1.1.4 Semi-supervised word sense disambiguation

As described in Section 1.1, the typical learning paradigms in WSD are supervised and unsupervised, and in some cases, they may rely on knowledge bases. However, there is a set of algorithms which lays between these two classes: they are semi-supervised algorithms. Generally, supervised learning requires a lot of data to reach a good generalisation. On the other hand, when small labelled data is available, it is a pity to leave it unused by moving to (knowledge-based) unsupervised learning. The strength of this class of algorithm arises when the labelled set size is not statistically significant to train an accurate fully-supervised classifier. In semi-supervised learning, only a small amount of hand-labelled (or automatically labelled) elements is needed; their information then will be propagated to the unlabeled part of the dataset. A typical example of semi-supervised learning applied in NLP is the one performed through bootstrapping. This set of methods can be split into two classes of algorithms: **self-training** and **co-training**.

A practical example of *self-training* is the **Yarowsky algorithm** [61]. Such an algorithm is fed with a small set of labelled data points in which at least one element per class (sense) exists, then a classifier is trained on it. After that, the trained classifier is used to assign a label to unlabeled data points. Eventually, the classifications which have a probability above a certain threshold are used to train a new classifier. This process is repeated for a fixed amount of steps, or until a convergence condition is reached, e.g. when the model parameters remain unchanged or the classifier is not able to assign a label with a probability over the defined threshold. This algorithm has been designed by Yarowsky relying on two main principles:

- **One sense per discourse:** which is *the observation that words strongly tend to exhibit only one sense in a given discourse or document*[2].
- **One sense per collocation:** which is *the strong tendency for words to exhibit only one sense in a given collocation*[3].

An approach similar to bootstrapping can be found in [66] which used a combination of a Naive Bayes classifier and the Expectation-Maximization algorithm. Even if in that paper it is used to for text classification it then became a standard algorithm to compare that has been used in a few papers, as in [67]. In [64], a first comparison between *self-training* and *co-training* is performed. However, a derivative of such paper which covers more algorithms can be found in [65] which adds to the list the *transductive* approach. In that paper, four algorithms have been proposed:

- **Co-training:** This method is based on the extraction of two different feature sets called *views*; with the assumption of being *conditionally independent*. For each view a classifier is trained, therefore, classification is performed on the unlabeled set and the most confident labelled elements are then added to the training set. By doing this, each classifier reinforces the other.
- **Smoothed co-training:** A modified version of co-training which introduces majority voting. The vote is between current classifiers and the ones that appeared during previous iterations. By doing this, typical performance decay over time is delayed.

[2] already defined in [62] but used for the first time in [61]

[3] introduced in [63]

- **Spectral Graph Transduction (SGT):** Graph-Transductive algorithms are based on different reasoning, in fact, rather than trying to infer an inductive, general classification rule, they focus just on the available observations.
  In SGT, a graph is built mapping both labelled and unlabeled elements as vertices and their similarities as edges. The labelling process is performed by partitioning the graph into two subgraphs and a different label is assigned to each subgraph. The partitions are selected accordingly to the cut that minimises a normalised-cut cost function.
- **Spectral Graph Transduction co-training:** A modified version of SGT which exploits the 'multiple views' principle of co-training.

These are some well known semi-supervised algorithms, however, there are a lot of examples that have been developed for some specific tasks. Like the one in [4], in which Label Propagation is performed and word sequences are exploited through LSTM networks. Or the one in [23] i.e. a semi-supervised game-theoretic approach for WSD (then moved to unsupervised in [24]) on which this thesis relies on. Such a framework will be discussed in the next chapters.

1.1.5 Visual Sense Disambiguation

A particular subdomain of WSD is the one related to verbs, this task is known as *Verb Sense Disambiguation*. For instance, considering the verb *run* the most common sense is the one related to the action of 'moving quickly':

- *If you want to break a record, you have to run faster.*
- *Every Saturday morning I run in the park.*

but there are other common senses, like the one related to 'machine operations':

- *The washing machine is running.*
- *With this tool, you can see the usage of processes that are running right now.*

or to cover a distance:

- *this train runs hundreds of miles every day*

all these senses share the same verb, but they have quite different meanings.

As for WSD, Verb Sense Disambiguation is an intermediate task which affects several domains, for instance in an image retrieval scenario, the goal is that the search engine being able to find the correct results in queries which contain ambiguous verbs and ideally that images get grouped by sense [76]. Another example is that a machine translation system being able to generate correct syntactic structures; or that in automatic summarisation the text is being misunderstood and so on.

The existing literature on Verb Sense Disambiguation is quite small with respect to the one related to general words. For instance, in [68], a smoothed maximum entropy system is used exploiting syntactical features based on linguistics. In [69] *Levin verb classes* are introduced and can be useful as priors for sense disambiguation. In [70], verb senses are clustered and mapped to sense inventory ones.

In Verb Sense Disambiguation the correct sense of a verb is inferred by exploiting the words surrounding it. In the field of Computer Vision there exists two tasks that have something in common with the WSD subtask: Action Classification (AC) and Human-Object Interaction (HOI). They exploit the identification of objects and entities in an image to infer either the action that is being performed or the correct verb that links those entities and objects. Even if there are some clear overlappings between WSD and AC/HOI, the latter do not take into account the ambiguity of verbs. The analogy between these tasks in NLP and CV fields may be exploited combining the features of both of them to improve the overall performance of the disambiguation system. For this reason, the task of *Visual/Multimodal Sense Disambiguation* (VSD) is introduced.

1.2 Related works

The majority of VSD systems that have been developed are designed to deal with nouns. One of the first approaches has been introduced in [8], which used a statistical model based on joint probabilities between images and words. In this case, the textual data is given by dataset captions, whereas image features are labels extracted with a region-labelling system. Moving to an unsupervised approach, in [6] they focused on *image sense discrimination* of web images and text. Spectral clustering has been applied to the combination of image features and textual features extracted from such data. A similar approach can be found in [73] which used co-clustering between textual domain and visual domain to discover multiple semantic and visual senses of a given noun phrase. In [7], they used *Latent Dirichlet Allocation* [42, 43]: a topic modelling technique; to discover a latent space by exploiting dictionaries definitions to learn distributions which represent senses. An interesting instance of multimodal disambiguation is the one proposed in [19]. In this case, the disambiguation targets are not the words of the sentence, but its overall meaning, for instance, given the following sentence: *Sam approached the chair with a bag*. There are two possible interpretations:

- *Sam is approaching the chair* ***carrying*** *a bag*
- *Sam is approaching the chair with a bag* ***on*** *it*

the goal of such a paper is to use images to solve those linguistic ambiguities. A similar task has been accomplished in [72].

All these systems perform quite good, however, they are mostly noun-oriented, having bad results for verb disambiguation. Nowadays, VSD for nouns can be easily performed exploiting State-Of-The-Art object detectors [8,73]. The first system able to perform fully verb-oriented VSD is the one introduced in *Disambiguating Visual Verbs* by *Spandana Gella, Frank Keller and Mirella Lapata* [2]. They proposed a modified **Lesk algorithm** [15] which deals with feature vectors of visual and textual data of both senses and inputs, using them as the definition and the context in canonical Lesk (a more detailed explanation will be given in Section 2.1.3). Another instance of multimodal disambiguation which deals with verbs is in [20]. Nevertheless, in such a paper, verb disambiguation is not the main task, since they use multimodal data for semantic frame labelling.

1.3 Goal

In this thesis, the Multimodal Sense Disambiguation problem is faced in a semi-supervised scenario relying on a game-theoretic framework. This approach has been chosen because, as described in Section 1.1.4 even if the amount of labelled information is much lower than in supervised approaches, a little supervision can help to boost the performance significantly, obtaining more accurate decision boundaries and higher classification accuracy with respect to unsupervised learning both in its pure or knowledge-based version [4]. Since the chosen semi-supervised learning approach is *transductive*, both labelled and unlabeled information is exploited to propagate the known labels to each unlabeled element. The selected transductive process is based on a game-theoretic model called *Graph Transduction Games* (GTG), defined for the first time in [21]. This game-theoretic model has already been successfully used for a WSD task both in a semi-supervised [23] and supervised [24] scenario and proved to perform better than other alternatives (like Label Propagation [22]) in semi-supervised tasks [3].

The main goal of this thesis is to map the Multimodal Sense Disambiguation dataset and the encoding pipeline implemented in the paper Gella et al. 2019: *Disambiguating Visual Verbs* [1]; into this GTG model in a way that each encoded feature point represent a node in the graph transduction algorithm. The dataset, the preprocessing of the data, the feature extraction, and performance evaluation are all based on the work described in the former paper. The VSD will be accomplished in both a unimodal and multimodal approach, taking into account both the image, its descriptions and tags. The experiments are performed on the recently published VerSe dataset [1], which is tailor-made for this task (whose composition is explained in Section 2.1.1).

1.4 Contribution

As described in Section 1.3, in this thesis, the pipeline used in Gella et al. 2019, is applied through GTG, so both approaches are gathering new results and applications. The main contributions are:

1. GTG is applied for the first time in a Multimodal Sense Disambiguation scenario.
2. Since the classification relies on Evolutionary Game Theory, players mutate their choices over time, whereas in Gella et al. 2019, just direct evaluation is performed.
3. The principle of label consistency is exploited, which means that the similarities among all the verbs are considered during inference and not only local information as in Gella et al. 2019 [1].
4. The results of previous state-of-the-art systems are outperformed using a labelled set which contains just two labelled data points per class. Anyway, the resulting performance improvement on the increase of labelled set size is documented.
5. In Gella et al. 2019, verb senses are encoded entities, so their creation is time-consuming, both for the annotation phase and for the data crawling phase. Using GTG, senses are only labels in semi-supervised learning. Thus, the creation of time-consuming hand-crafted queries and the consequent data crawling is avoided.

This thesis is composed as follows: in Chapter 2, all the techniques used in Gella et al. 2019, are illustrated. Describing the characteristics and limits of the approach. After that, the theoretical foundations of GTG are given, moving then to the implementation details. In Chapter 3, the proposed disambiguation algorithm is then compared to the former one in all its configurations and their results are then examined in Chapter 4. Eventually, conclusions and future works on the topic will be discussed in Chapter 5.

Chapter 2

Visual Verb Sense Disambiguation

The pipeline of this approach is composed of four main steps:

1. Cues extraction
2. Features embedding
3. Graph construction
4. Verb-sense assignment

In order to understand properly the strengths of Graph Transduction Games with respect to the algorithm proposed in Gella et al. 2019 (that covers the first two steps), both approaches are explained is explained in Section 2.1 (Gella et al. 2019) and Section 2.2 (GTG).

2.1 Original approach

In Gella et al. 2019 there are two VSD methods:

- The system is fed with an input image and a target verb to disambiguate. The features extracted from the image/object labels/captions are used to perform the task.
- The system is fed just with an image, verbs are predicted using either a multilabel classification algorithm or Multiple Instance Learning (MIL). After that the features extracted from the image/object labels/captions are used to perform the task.

in this thesis, only the former approach is explored.

2.1.1 Dataset

As pointed out in Section 1.1.5 there is a connection between Verb Sense Disambiguation and Action Classification (AC). For this reason, a good example of data that could have fit this new task is the one extracted from AC datasets. However such datasets usually have a small size or they are domain-oriented. Moreover, they lack generalisation and contain ambiguous labels. For this reason, a new dataset for Visual Sense Disambiguation has been created: the *VerSe* (Verb Sense) dataset.

VerSe is built as a combination of Microsoft COCO (Common Objects in Context) [13] and TUHOI (the Trento Universal Human-Object Interaction dataset) [12]. COCO is not an action recognition dataset, however, its captions contain verbs that can be used for semantic role tagging. On the other hand, the majority of TUHOI action categories occur only once. For this reason, VerSe is the result of the combination of some limited parts of COCO and TUHOI.
The resulting dataset is composed of 3518 images and 90 verbs. In COCO every image is described by multiple captions and there is an object label which tags each object that appears in it. For instance, for the action which takes place in Figure 2.1 the captions are:

- *young woman in orange dress about to serve in tennis game, on blue court with green sides.*
- *a woman taking a swing at a tennis ball*
- *a girl dressed all in orange hitting a tennis ball.*
- *a tennis player getting ready to serve the ball.*
- *a woman swinging a tennis racquet at a tennis ball.*

and the object labels are *person, sports ball, tennis racket, chair*.

The textual part of TUHOI is slightly different, it does not contain natural like descriptions, but text fields which can be resumed in verb-object pairs. For instance, for the action which takes place in Figure 2.2, the available captions are:

- *sing/none/use — microphone*
- *play/play/play — saxophone*

in this case, multiple verbs fit with the object *microphone*, thus, three different sentences are generated. Whereas, for the object *saxophone*, only the verb *play* is compatible with it, and it is repeated three times (having three equal sentences). When TUHOI captions are used in VerSe, they are slightly modified: the noun *person* is prepended each pair, having a triple person-verb-object (e.g. *person — use — microphone*).
Regarding object labels, they are made using the elements which appear in the object fields, so for Figure 2.2, the object labels are *microphone, saxophone*.

Figure 2.1: A sample image from COCO dataset.

Figure 2.2: A sample image from TUHOI dataset.

In VerSe, every image is related to one or more verbs. For each of such <image, verb> pairs there is a sense identifier. Moreover, the dataset is provided with a list of verb senses. In fact, for each verb, there is a group of identifiers (one for each sense) and for each verb sense, there is a typical dictionary definition and some examples of use (such features are extracted from OntoNotes). Furthermore, every verb sense is paired with a list of web queries. Such queries, if inserted in a search engine, should fetch a list of images which matches their sense. Lastly, there is a classification of verbs in Levin classes (motion and non-motion verbs), since experiments have been performed taking into account such a separation.

The VSD method introduced in this thesis is run only on VerSe dataset, this is done mostly to have a real performance comparison with Gella et al. 2019 and because, as described previously, there is a lack of image datasets that could fit in this task.

2.1.2 Setups

VSD is performed through 7 main setups, by analysing:

- captions (C)
- object labels (O)
- captions combined with object labels (O+C)
- images (CNN)
- images combined with captions (CNN+C)
- images combined with object labels (CNN+O)
- images combined with object labels and captions (CNN+O+C)

this is done to understand which feature impact more on the disambiguation. Regarding textual annotations, there are two available setups:

- **GOLD captions:** The original captions and object labels provided with VerSe are used as information to disambiguate (as expected).
- **PRED captions:** Captions are generated using a tool which describes images with sentences. Object labels are extracted using an image classifier.

this distinction is done to project the results of the paper to a harder scenario which simulates real-life conditions i.e. having an image without captions. GOLD captions are written by human annotators, whereas in PRED, a state-of-the-art image descriptor is used, which would be a mandatory choice in a real application.

PRED annotations generation: Captions In PRED setup, image captions are generated using *NeuralTalk*[1] a tool developed in *Karpathy and Fei-Fei, 2015* [32]. This system generates natural language descriptions of images and their regions. This process is carried out by combining Convolutional Neural Networks with bidirectional Recurrent Neural Networks. The training is performed by feeding the system with an image and its captions. There is a first model whose goal is to associate sentence entities with their visual regions. Afterwards, Such relations are used as training data for the bidirectional RNN, whose goal is to learn to generate captions.
This tool has been improved by the Google brain team [31], the structure is almost the same, but they did some modifications like replacing RNN with LSTM.

PRED annotations generation: Object labels In the PRED setup, object labels are extracted using state-of-the-art image classifier. At the time of the first article drafting (published in 2016 [2]), the most advanced neural networks for image classification were the set of VGGs. In this case, VGG16-layers has been used (a more detailed overview of VGGs can be found in Paragraph 2.1.4.1). VGGNet does not perform object detection, therefore they used a trick to extract object labels: they extracted the probabilities of the SoftMax activation layer, and set a threshold of 0.2, the classes that had a probability higher than the threshold were used as object labels. When none of these classes satisfies the requirements, the class with the highest probability is used. With this technique, multiple classes/objects labels per image are obtained.

2.1.3 Lesk Algorithm

Gella et al. 2019 implementation is based on a modified version of the *Lesk algorithm* [15]. Lesk is a typical algorithm used on unsupervised-knowledge-based WSD algorithms, its most famous version is the *Simplified Lesk Algorithm* [16]. The main idea behind it is to count the overlapping words between the target word neighbourhood i.e. *context* and the sense definition in the sense inventory i.e. definition; The higher the number of overlapping words, the more likely to be the correct sense.

- **Definition:** the words which appear in the dictionary-like definitions of the word sense and (if available) their examples of use.
- **Context:** the words which occur nearby the target word. They are selected according to a certain window size. It can select one word on the right and one word on the left, two words on the right and two words on the left and so on; it can also be asymmetrical.

So, given a knowledge base such as a dictionary or an ontology like WordNet or OntoNotes, a for each sense definition of the target word to disambiguate, a list the

[1]https://github.com/karpathy/neuraltalk

list of words which compose them is extracted, after that, the words appearing in the the sentence which fit in a context windows of size N, are inserted in another list. Then, the sense whose list has more overlappings with the context list is returned as the more proper word sense. The behaviour of the algorithm is summarised in the following pseudocode (taken from [78]):

```
max-overlap ← 0
context ← set of words in sentence:

for each sense in senses of word do
    signature ← set of words in the gloss
                                    and examples of sense
    overlap ← COMPUTEOVERLAP(signature, context)
    if overlap > max-overlap then
        max-overlap ← overlap
        best-sense ← sense
end
return best-sense
```

where `word` and a `sentence` are the input parameters of the algorithm. The `COMPUTEOVERLAP` function computes the number of word which appear both in the word signature and the word context.

bank-1	definition	*a financial institution that accepts deposits and channels the money into lending activities*
	examples	*he cashed a check at the bank*, *that bank holds the mortgage on my home*;
bank-2	definition	*sloping land (especially the slope beside a body of water)*
	examples	*they pulled the canoe up on the bank*, *he sat on the bank of the river and watched the currents*;

Table 2.1: Dictionary definitions of the word *bank*. (example taken from [78]).

Here it is an example [78] of usage the algorithm on disambiguating the word *bank* in the following sentence and the definitions in Table 2.1:

> *The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.*

the definition and the examples of the first sense of the word *bank* in the dictionary has overlappings with two words i.e. *deposit* and *mortgage* whereas for the second sense there are no overlappings, hence the former is selected.
This Lesk version and can be formalised in the following function [1]:

$$\hat{s} = \arg\max_{s \in \mathcal{S}(v)} \Phi(s, v, \mathcal{D}) = |\text{context}(v) \cap \text{definition}(s, \mathcal{D})| \tag{2.1}$$

where context(v) function represents the nearby words of the target word v and definition$(s, \mathcal{D})$ function represents the words which occur in the definition of sense s in the dictionary $\mathcal{D}$.

As pointed out previously, there are several implementations of the Lesk algorithm, some of them rather than simply count overlappings, they weight them in different ways. The original version of Lesk [15] is quite different since it does not count the overlappings with target word context, but the overlappings with its definition.
For instance, given the sentence *pine cone* and the sense definitions of Table 2.2, for the word *cone*, the Lesk algorithm will return the third sense entry. The reason of this selection is that there are two overlapping words between the third sense of *cone* and the first sense of *pine* i.e. *evergreen* and *tree*.

Verb	**Sense ID**	**Definition**
pine	1	*kinds of evergreen tree with needle-shaped leaves*
pine	2	*waste away through sorrow or illness*
cone	1	*solid body which narrows to a point*
cone	2	*something of this shape whether solid or hollow*
cone	3	*fruit of certain evergreen trees*

Table 2.2: Dictionary definitions of the words *pine* and *cone* (example taken from [78]).

Now that the main idea behind the set of Lesk algorithms and the version introduced in Gella et al. 2019 is described. In such an implementation, the algorithm is run on the encoded feature vectors of context and definition rather than their textual counterpart. It is formalised as follows:

$$\hat{s} = \arg\max_{s \in \mathcal{S}(v)} \Phi(s, v, i, \mathcal{D}) = i \cdot s \tag{2.2}$$

where s represents the encoded features of the dictionary definition/images from $\mathcal{D}$ and i represents the encoded features of the input image in which the verb v occurs. In Equation 2.2, the vectors i and s are assumed to be unit-normalised, so the dot product $(\cdot)$ operator is intended as a *cosine distance similarity* i.e.

$$\cos\theta = \frac{x \cdot y}{\|x\| \cdot \|y\|}$$

where x and y are two input vectors and the norm operator is the Euclidean norm:

$$\|x\| := \|x\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}$$

where $x = (x_1, \ldots, x_n)$. The sense with the most similar feature vector with the input one is selected.

2.1.4 Features encoding

As pointed out in Section 2.1.2, the available data can be used in several setups, therefore, different encodings can be used, they can be split into three main categories: *textual*, *visual* and *multimodal*. In this model, encoded data is represented as unit-normalised feature vectors to exploit the cosine similarity measure.

2.1.4.1 Visual features

In order to encode images, a model able to extract their features is needed. A typical algorithm like *HOG* (Histogram of Oriented Gradients) [74] or *SIFT* (Scale Invariant Feature Transform) [75] could be used. However, in the last few years, the power of deep learning is outperforming almost every other method, for this reason, an approach of this type has been preferred. In Gella et al. 2019 the most powerful image classifier at the time has been selected: *VGGNet* [44].

VGGNet VGGNet scored promising results in ILSVRC 2014 (Large Scale Visual Recognition Challenge) both in classification and localisation tasks. Such network is characterised by the substitution of the big convolutional filters used in *AlexNet* [45] with stacks of small (3×3) convolutional filters, resulting in a network with the same receptive field but with more non-linearities and fewer parameters.

VGGNet is provided in two variants:

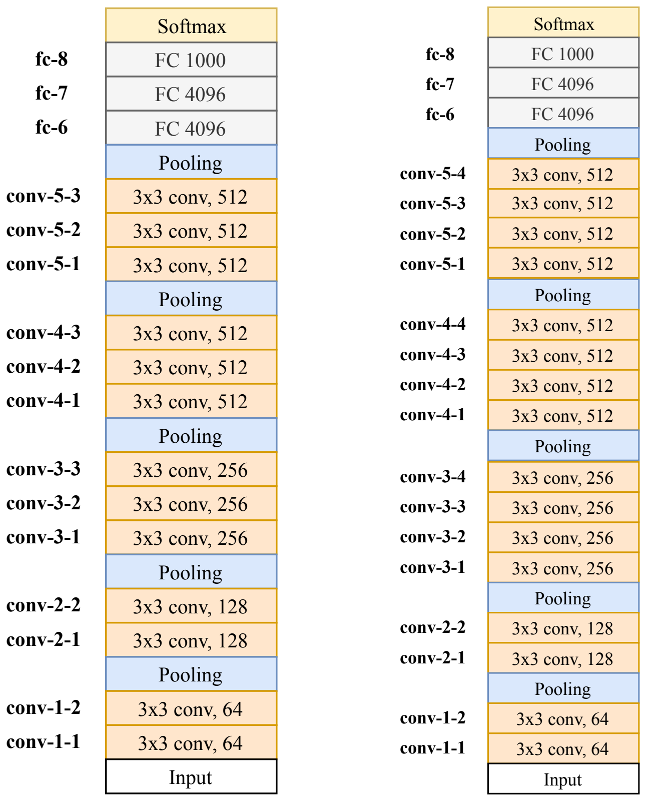
- **VGG16:** (Figure 2.3a) it is composed of two stacks of two convolutional filters and three stacks of **three** convolutional filters. On top of them, there are three fully-connected layers. Each stack is separated by a max-pooling layer.
- **VGG19:** (Figure 2.3b) it is composed of two stacks of two convolutional filters and three stacks of **four** convolutional filters. On top of them, there are three fully-connected layers. Each stack is separated by a max-pooling layer.

with respect to VGG16, VGG19 is slightly more accurate, however, it requires much more memory because of the greater number of parameters to train.

Encoding VerSe images In Gella et al. 2019, has been selected as feature extractor, a VGG16 network pre-trained on ILSVRC 2012 object classification dataset. The *FC-7 layer* of VGG16 is used as feature representation. The resulting encoding is a vector composed of 4096 elements (such vector is then unit-normalised as described in Section 2.1.3)

2.1.4.2 Textual features

Gella et al. 2019 approach is designed to be run with multimodal data, however, the encoding of text and images is independent (in the first stage of the process), hence a representation for textual data has been selected: *word2vec* [33].



(a) VGG16 (b) VGG19

Figure 2.3: VGGNet architectures

word2vec In NLP, words or sentences are usually encoded in vectors composed of real numbers. Such vectors are called *word embeddings*, they can capture information about the semantics of encoded-words and their context.
A typical encoding in NLP is *co-occurrence matrix* which represents the frequencies of words which appear nearby; or *tf-idf* which is its weighted version. Nevertheless, these approaches produce sparse and long vectors with a dimensionality corresponding to the size of the vocabulary. Due to the computational effort, and the easier representation of relationships between words, it might be better to have short and dense vectors [78]. A method able to produce vectors which satisfy such requirements is *skip-gram with negative sampling*; it is often called *word2vec*, from the name of the package which provided it.
Such algorithms aim to redesign the idea which stands behind word encoding. The intuition is to move the core of the algorithm from word frequencies to a binary classification task of the likelihood of a word occurring close to another one. Nevertheless, such a task is only a medium to learn a set of weights able to represent the text dictionary and map it into a vector space.

Therefore, the main idea behind skip-gram is [78]:

1. *Treat the target word and a neighbouring context word as positive examples.*
2. *Randomly sample other words in the lexicon to get negative samples*
3. *Use logistic regression to train a classifier to distinguish those two cases*
4. *Use the regression weights as the embeddings*

So, the classifier task is to decide whether a word c occurring near the target word t is a *context word* by computing such probability; this is done by using the similarity between them. Such result is then fed into an activation function (such as sigmoid or logistic) to convert the real value into a probability. The goal is to maximise the similarity (dot product) of the target word with positive samples and minimise it with negative samples. In this process, the classifier will learn an embedding for the target word and an embedding for its context words. The optimisation process is carried out with gradient descent to maximise the objective function.

Even if the true purpose of this process is to learn word embeddings, the classification task on which it relies on can be performed by exploiting neural networks. This approach provides two different models [33]:

- **Continuous Bag of Words (CBOW):** Each word in a sentence is predicted given its context words.
  For instance, given the sentence: *the quick brown fox jumps over the lazy dog* and the context words *quick, brown, jumps, over*; the target word to predict is *fox*.
- **Skip-Gram:** The context words of the target word are predicted.
  For instance, given the sentence: *the quick brown fox jumps over the lazy dog* and the target word *fox* the context word to predict are: *quick, brown, jumps, over*.

by doing this, the trained weight matrix can be seen as a lookup table for word vectors.

CBOW Continuous Bag of Words algorithm can be split into the following steps [33, 79]:

1. Each word $w_{I,1}, \ldots, w_{I,C}$ in the window is encoded using one-hot representation, in which the vector element in the position of the word in the dictionary is set to *one* and *zero* to every other position.
2. Such vectors are then multiplied by the weight matrix[2] W and averaged. Such a matrix is randomly initialised at the beginning of the process.

[2]The weights are the ones which link the input layer to the hidden layer

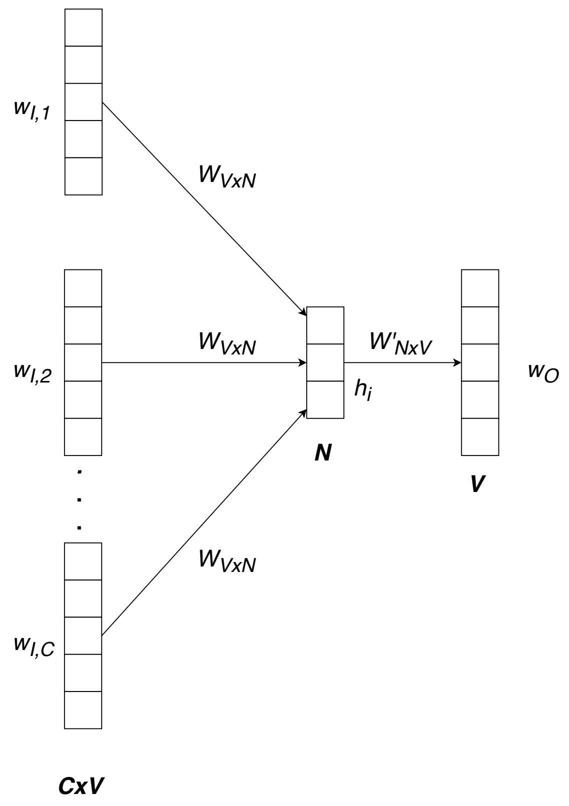


Figure 2.4: CBOW neural network representation.

3. The resulting vector is passed through a *linear* activation function.
4. The output of the activation function is then multiplied by another weight matrix[3] W'.
5. A SoftMax function is used to compute the probabilities of the next occurring word.
6. The parameters are trained through gradient descent trying to predict the next occurring word. The loss function is the cross-entropy function:

$$L = -\log p(w_O \mid w_{I,1}, \ldots, w_{I,C})$$

   where w_O is the output target word to predict and $w_{I,1}, \ldots, w_{I,C}$ are the context input word.

all this process can be resumed with the neural network in Figure 2.4.

[3]The weights are the ones which link hidden layer to the output layer

Skip-Gram The Skip-Gram model can be seen as a reverse CBOW. Starting from the conceptual idea behind it: in Skip-Gram the goal of the model is to compute the probability of each word in the vocabulary to appear in a random position nearby (considering a certain window) the target word. The Skip-Gram algorithm can be split into the following steps [33, 79]:

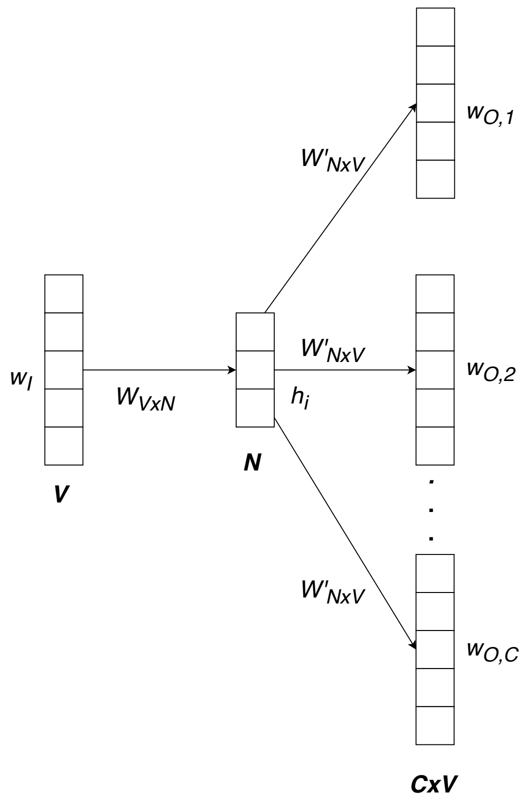


Figure 2.5: Skip-Gram neural network representation.

1. The target word w_I is encoded using one-hot representation.
2. The input vector is then multiplied by the weight matrix W.[4]
3. The resulting vector is passed through a *linear* activation function.
4. The output of the activation function is then multiplied by another weight matrix W'.
5. In the output layer, there is a multinomial distribution for each word 'placeholder' in the context window.

[4]Note that since the vector is one-hot encoded and no mean operations are performed, the result of the multiplication is the transposition of the i-th row of W.

6. The parameters are trained through gradient descent trying to predict the next occurring word. The loss function is the cross-entropy function:

$$L = -\log p(w_{O,1}, w_{O,2}, \dots, w_{O,c} \mid w_I)$$

   where $w_{O,1}, w_{O,2}, \dots, w_{O,c}$ are the output context words to predict and w_I is the target input word.

all this process can be resumed with the neural network in Figure 2.5.
The differences of use between the two models are that Skip-Gram works well with a small amount of data and is able to capture rare occurring words well. On the other hand, CBOW is faster and has better representations for more frequent words.

word2vec parameters and vector semantics A parameter in word2vec models is the *dimensionality*. According to the results provided in the original paper [33], the higher the dimensionality, the higher the accuracy, although, there is a point in which a dimensionality increase implies a very small accuracy gain. To have better growth, the dataset size must be increased as well. In this case, the more the training data, the better the accuracy. A typical compromise on embeddings dimensionality is 300 since is the point in which the accuracy starts capping if the data size remains the same.
Vectors dimensionality is a parameter that needs to be tuned, nevertheless, it has to be kept in mind that longer vector are harder to keep in memory and they need more training time to learn weights.

As pointed out at the beginning of the paragraph, Skip-Gram and CBOW models produce embeddings which can capture semantic properties of words. An ideal model should always be able to respect semantics like in the following examples (which are visualised in Figure 2.6):

$$\textit{Paris} - \textit{France} + \textit{Italy} = \textit{Rome}$$
$$\textit{King} - \textit{Male} + \textit{Female} = \textit{Queen}$$

In Table 2.3 there is a list of relationships kept in a Skip-gram model trained on 783M words. Several relationships have been caught i.e. country — capital, adjective — superlative, president — country, company — product, etc. In this case, the model scored a 60% exact-match accuracy. According to authors [33], the model would have produced better results by increasing the dimensionality of embeddings (these were 300-dimensional vectors) and perform the training on a larger dataset. Several techniques and tricks have been used to improve word2vec algorithms like frequent words subsampling and negative sampling. Furthermore, some other models have been built on top of word2vec, like *GloVe* [80] and *FastText* [81].

[5] The table as been taken from the original paper *Efficient Estimation of Word Representations in Vector Space* [33]

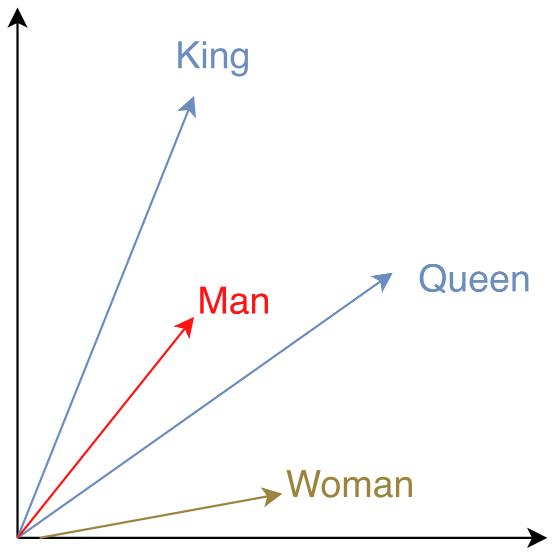


Figure 2.6: word2vec semantics.

Encoding VerSe descriptions As pointed out in Section 2.1.2, in Gella et al. 2019, the unimodal encoding of image descriptions is used in three setups:

- **Captions (C):** Each word of the caption is fed into a word2vec Skip-gram model.
- **Object labels (O):** All the words which compose object labels are fed into a word2vec Skip-gram model.
- **Combined (O+C):** The caption and the object labels are concatenated into a single string, each word in it is fed into a word2vec Skip-gram model.

Relationship	Example 1	Example 2	Example 3
France — Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big — bigger	small: larger	cold: colder	quick: quicker
Miami — Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein — scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy — France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper — Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi — Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft — Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft — Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan — sushi	Germany: bratwurst	France: tapas	USA: pizza

Table 2.3: Examples of word relationships learnt in the embedding[5].

the model will produce an embedding vector for each input word. Such vectors are unit-normalised, and an overall average is computed between each output vector of the caption/labels/both (depending on the setup) and unit-normalised once again. By doing this, for each configuration, there is an embedding which represents the textual data of the image, therefore, an embedding for each image (Figure 2.7).

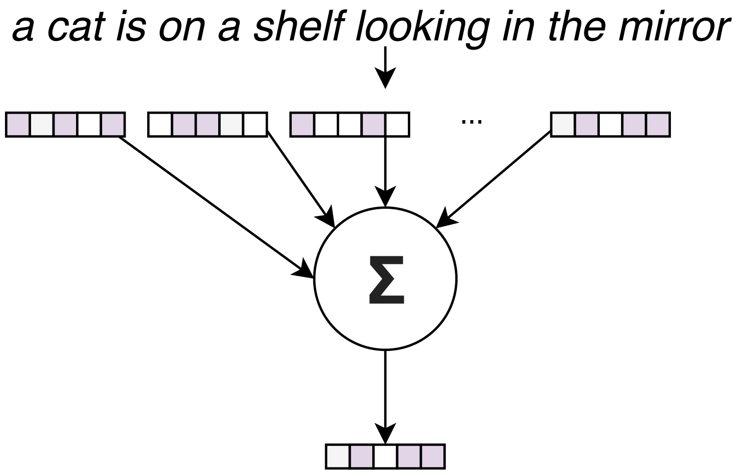


Figure 2.7: word2vec encoding pipeline.

Such encoding is performed using a Gensim model [26] which produces vectors in a 300-dimensional space. It has been pre-trained on Google News corpus dataset, composed of about 100 billion words.

2.1.4.3 Multimodal features

Gella et al. 2019 is a paper which aims to deal with multimodal data in which three techniques are used to combine the feature vectors of Section 2.1.4.1 and Section 2.1.4.2:

- Concatenation
- Canonical Correlation Analysis
- Deep Canonical Correlation Analysis

also in such cases, for each combination technique, three setups are evaluated: *image with captions, image with object labels, image with object labels and captions*; as pointed out in Section 2.1.2.

Features concatenation In this encoding the 300-dimensional textual feature vector is appended to the 4096-dimensional visual-feature vector, resulting in a 4396-dimensional vector which captures both traits.

(Deep) Canonical Correlation Analysis *Canonical Correlation Analysis (CCA)* [82] is a statistical method which receives in input two vectors of random variables X and Y which correlate between them. The goal is to train a model able to transform vectors through a linear combination of them which maximises their correlation. By doing this the two vectors are projected in a new vector space in which they are correlated. CCA has already been used to combine textual and visual data, producing good results for an image retrieval scenario [83].

On the other hand, *Deep Canonical Correlation Analysis (DCCA)* [84] is a non-linear version of CCA, this is done by passing data through two deep neural networks in which the output layers are maximally correlated.
DCCA has already been used for combining textual and visual data [85], overcoming the results of linear CCA and its kernel versions (KCCA).

Encoding VerSe with (Deep) Canonical Correlation Analysis In Gella et al. 2019, CCA and DCCA are used to project textual and visual data into a joint latent 300-dimensional space. The resulting vectors are then merged through a convex combination as follows:

$$i^m = \lambda i^{t'} + (1 - \lambda) i^{c'}$$

where i^m is the output multimodal representation, $i^{t'}$ is the projected image feature vector (i^t), $i^{c'}$ is the projected textual feature vector (i^c) and λ is balancing parameter which weights the relevance of the vectors.

The models are trained using the textual features (Section 2.1.4.2) of descriptions provided with COCO and Flickr30k [14] datasets and the visual features (Section 2.1.4.1) of their respective images. The best scores have been obtained with a balancing parameter $\lambda = 0.5$.

2.1.5 Sense Encoding

As described in Section 2.1.3, Gella et al. 2019 algorithm relies on Lesk algorithm. It needs an input-image representation (context), whose encoding has already been illustrated in Section 2.1.4 and a set of sense definitions. In this section it is show how such definitions are encoded in vector-form in order to compute the similarity with the context and infer the most likely sense for the verb; therefore, in this setup, there is a vector for each available sense.

Also in this case, different encodings can be used, which can be split into three main categories: *textual*, *visual* and *multimodal*.

2.1.5.1 Visual features

In a standard Lesk algorithm usage, the textual definition of senses are extracted from dictionaries and knowledge bases, however, in this case, such information would not fit with image data. Visual definitions can be retrieved from ImageNet [17]: a wide knowledge base for images. Nevertheless, such data is related to nouns, and there is a lack of information regarding verbs. For this reason, a technique to have an image-feature-based representation of verb senses has been designed.

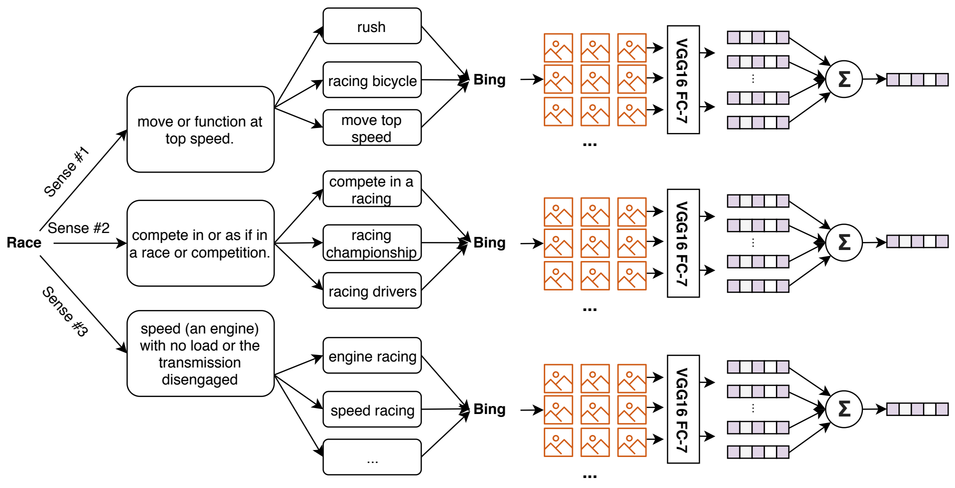


Figure 2.8: Visual-sense encoding pipeline.

In Gella et al. 2019, a plethora of sense-related images has been fetched from the web. The pipeline can be found in Figure 2.8 and can be resumed as follows:

1. For each verb sense s in the dictionary, at least three queries that should return images related to it have been defined by human annotators.

2. For each query $q \in \mathcal{Q}(s)$ the topmost fifty images returned by Bing search engine have been fetched.

3. Each image $i \in \mathcal{I}(q)$ is converted into a feature vector c_i using the FC-7 layer of VGG16 as described in Section 2.1.4.1.

4. The fetched images of the same sense are averaged as follows:

$$s^c = \frac{1}{n} \sum_{q_j \in \mathcal{Q}(s)} \sum_{i \in \mathcal{I}(q_j)} c_i$$

   where s^c is the vectorial representation of the sense and n is the total number of images fetched for s.

2.1.5.2 Textual features

The approach used for textual encoding for senses is very similar to the one used for the descriptions in the dataset. In this case, a list of verb-sense definitions and usage examples has been extracted from OntoNotes (as described in Section 2.1.1). Such data is then encoded into a vectorial representation i.e. for each verb sense, each word composing its definition and usage examples is fed to the same word2vec Skip-gram model used in Section 2.1.4.2, such vectors are then averaged and normalised, resulting in a single 300-dimensional vector for each verb sense.

2.1.5.3 Multimodal features

The multimodal encodings used for senses are the same used in Section 2.1.4.3. Therefore, given the textual representation s^t and the visual representation s^i of the sense, they are can be combined with:

- Vector concatenation
- Canonical Correlation Analysis
- Deep Canonical Correlation Analysis

in the two latter cases, the combination equation of the two projected vectors is the following:

$$s^m = \lambda s^{t'} + (1 - \lambda)s^{c'}$$

where s^m is the output multimodal representation, $s^{t'}$ is the projected image feature vector, $s^{c'}$ is the projected textual feature vector and λ is balancing parameter which weights the relevance of the vectors.

2.1.6 Unsupervised Visual Verb Disambiguation

Once the encoding has been performed, there are the following elements available:

- Three possible textual representations i^t for each image (one with captions, one with object labels and one with both of them).
- A visual representation i^c for each image.
- Nine possible multimodal representations i^m, three for each setup (concatenation, CCA, DCCA) for each image (one with the image and captions, one with the image and object labels and one with both of them).
- A textual representation s^t for each verb sense in the dictionary.
- A visual representation s^c for each verb sense in the dictionary.
- Three (one for setup: concatenation, CCA, DCCA) possible multimodal representations s^c for each verb sense in the dictionary.

According to the setup, there is a set of image representations and sense representations. As defined in Section 2.1.3, the disambiguation algorithm is the following:

```
max-similarity ← 0
context ← set of features from image:

for each sense in senses of verb do
    sense_definition ← set of features from
                            sense inventory  of sense
    similarity ← COSINESIM(sense_definition, context)
    if similarity > max-similarity then
        max-similarity ← similarity
        best-sense ← sense
end
return best-sense
```

where the input parameters of the algorithm are the `image`, and the `verb`. And the function `COSINESIM` computes the similarity between the two vectors.

2.2 Graph Transduction Games for VSD

In Gella et al. 2019 the typical WSD task has been enchanced. Introducing a new dataset, and using state-of-the-art encoding (at the moment of writing the first version [2], language models like *ELMo* [86] and *BERT* [87] and image classifiers like *ResNet* [88] were not yet available[6]) which used some feature-combination that got interesting results in the multimodal field.

In this thesis, the pipeline provided in Gella et al. 2019 has been inserted in a more dynamical space which relies on *game theory* and is used in a semi-supervised setup. As each semi-supervised learning setup, this approach stays in the middle between a fully supervised algorithm and the one proposed in Gella et al. 2019. However, the main goal here is to overcome the limits of the former approach.

The first of them is the static nature of the modified Lesk application. When selecting the verb sense which is more similar to the input image, the operation is performed in an independent manner, which does not take into account neither all the (possible) previous classifications nor the information of other embeddings. This is a waste of information which could be exploited to improve the next results. Moreover, the classification is a one-step operation which does not iterate since it does not rely on evolving states kept by the algorithm.

The second limit consists of the encoding of a sense-representing entity. Since each classification is independent, there is a strong assumption about the proper representation of the encoded sense. If this task can be easily accomplished for the textual data, it is quite hard to replicate for the visual features since it is required a lot of data to ensure a reliable representation. The fetching operation is very time-consuming and probably not replicable, due to the possible modification of search engine internals.

In this thesis, these two disadvantages are being overcome by using a transductive algorithm which relies on Evolutionary Game Theory which eliminate the concept of sense-representing entity, treating senses only as class labels in a classification task.

2.2.1 Semi-supervised learning with Graph Transduction Games

Graph Transduction Games (GTG) [21] is a graph-based approach to perform semi-supervised learning. Thus, as described in Section 1.1.4 the information coming from both labelled and unlabeled data points is leveraged to perform classification. In this class of algorithms, the data is modelled as a graph $G = (V, E, w)$ whose vertices are the observations in the dataset and edges represent similarities among them. The set of nodes is defined as $V = L \cup U$, where $L = \{(f_1, y_1), \dots, (f_l, y_l)\}$ is the set of labelled observations, whereas $U = \{f_{l+1}, \dots, f_n\}$ is the set of unlabeled ones, with f_i being feature vectors and $y_i \in \{1, 2, \dots, m\}$ being labels. Weights w are a general measure of consistency among a pair of observations assigned to each edge $(u, v) \in E$, which can be given in advance or computed

[6]However, nowadays VGGNet is still a widely used feature extractor.

using the features of the observations. This representation is then exploited to perform inference by *propagating* the labelling information of each node to its direct neighbours. GTG algorithm relies on the *cluster assumption*, which is composed of two principles:

1. *data points that are close to each other are expected to have the same label.*
2. *data points in the same cluster are expected to have the same label.*

In this transductive classification algorithm, the graph represents a context in which a *noncooperative multiplayer game* is performed. Data points are mapped to the players, which play games with their neighbours choosing among a set of action (strategies) which represent their labels. They receive a payoff based on the similarities among the neighbours along with the strategies they have selected. This game is run multiple times until a point of convergence is reached, namely the *Nash Equilibrium* [37], which provides a well-founded way of consistent labelling for the unlabeled data points. Since this is a semi-supervised setting, the strategies of labelled players are already set to the ones corresponding to their actual label.

2.2.1.1 Framework definition

Noncooperative Game Noncooperative Game Theory aims to analyse the individual strategic behaviour (*games*) of anonymous agents (*players*). The goal of each player is to interact with the game in a way which its own utility (*payoff*) is maximised. Players can play the game by selecting actions (*pure strategies*) provided by it. When a strategy is played, a payoff is returned and it is dependent on the choices of each player in the game. A Multiplayer Noncooperative Game can be defined as:

$$G = (\mathcal{I}, S, \pi)$$

where:

- $\mathcal{I} = \{1, \ldots, n\}$ is the *set of players* ($n \geq 2$).
- S is the *joint strategy space* given by the Cartesian product of the individual pure strategy sets $S_i = \{1, \ldots, m_i\}$ (with $i \in \mathcal{I}$).
- $\pi : S \to \mathbb{R}^n$ is the *combined payoff function* which assigns a payoff $\pi_i(s)$ to each pure strategy $s \in S$ of each player $i \in \mathcal{I}$.

Mixed strategies and mixed profiles *A* mixed strategy *is a probability distribution over its pure strategy set S_i.* It can be formalised as $x_i = (x_{i1}, \ldots, x_{im_i})^T$ where each element x_{ih} represents the probability that the player $i \in \mathcal{I}$ plays its h-th pure strategy. Mixed strategies are intended to model the uncertainty in choosing one pure strategy over another. Mixed strategies lie in the standard simplex of the m_i-dimensional Euclidean space $\mathbb{R}^{m_i}$:

$$\Delta_i = \left\{ x_i \in \mathbb{R}^{m_i} : \sum_{h=1}^{m_i} x_{ih} = 1, \text{ and } x_{ih} \geq 0 \text{ for all } h \right\}$$

Taking into account the mixed strategies of all players $i \in \mathcal{I}$ results into a *mixed strategy profile*, it is defined as $x = (x_1, \ldots, x_n)$ with $x_i \in \Delta_i$ and represents the state of the game in a certain moment. Mixed strategy profiles lie in the *mixed strategy space:*

$$\Theta = \times_{i \in \mathcal{I}} \Delta_i$$

Pure strategies are probabilistically equivalent to *extreme mixed strategies*. They are denoted as e_i^h, with 1 at position h and 0 elsewhere.

2.2.1.2 Payoff functions

To evaluate the best choice for each player, a tuple of *payoff functions* $u = (u_1, \ldots, u_n)$ s.t. $u : \Delta^{n \times m} \to \mathbb{R}^n_{\geq 0}$ is defined. Such payoffs quantify the gain that each player obtains given the actual configuration of the mixed strategy space.

Let $z = (x_i, y_{-i}) \in \Theta$ denote a strategy profile in which player i plays strategy x_i, whereas every other player $j \in \mathcal{I} \setminus \{i\}$ plays on the strategy profile $y \in \Theta$. The notation $z = (x_i, y_{-i}) \in \Theta$ is used to indicate a strategy profile in which player i plays strategy x_i and every other player $j \in \mathcal{I}$ *s.t.* $i \neq j$ plays on the strategy profile $y \in \Theta$. The expected value of the payoff of player i is the following:

$$u_i(x) = \sum_{s \in S} x(s) \pi_i(s) = \sum_{k=1}^{m_j} u_i(e_j^k, x_{-j}) x_{jk}$$

where $u_i(e_j^k, x_{-j})$ represents the payoff gained by player i when player j selects its extreme mixed strategy $e_j^k \in \Delta_i$.

Payoffs in GTG In this graph transductive setup, it is assumed to play in a polymatrix game [40] where only pairwise interactions are allowed and the payoffs associated to each player is given by the sum of the payoffs gained from each game played with one of its neighbours. With this assumption, the payoff function can be cast to:

$$\pi_i(s) = \sum_{j=1}^{n} A_{ij}(s_i, s_j)$$

where $A_{ij} \in \mathbb{R}^{m \times m}$ is the *partial payoff matrix* between player i and player j. In particular, $A_{ij} = I_m \times \omega_{ij}$ where I_m is the identity matrix and ω_{ij}, is the similarity between player i and j. Therefore, the payoffs can be computed as:

$$u_i(e_i^h, x_{-i}) = u_i(e_i^h) = \sum_{j \in U} (A_{ij}x_j)_h + \sum_{k=1}^{m} \sum_{j \in L} A_{ij}(h,k) \tag{2.3}$$

$$u_i(x) = \sum_{j \in U} x_i^T A_{ij} x_j + \sum_{k=1}^{m} \sum_{j \in L} x_i^T (A_{ij})_k \tag{2.4}$$

where Equation 2.3 is the payoff received by the player i when playing strategy h and Equation 2.4 is the expected payoff for the player i, for its mixed strategy x_i. The goal of each player is to pick a strategy which leads to a payoff greater or equal than the expected ones.

2.2.1.3 Nash equilibrium

A central concept in Game Theory is *Nash equilibrium* which is a strategy selection performed by players such that no player, has an incentive to deviate from his current strategy because a unilateral change cannot increase his payoff. Such a concept is formalised as follows:

$$u_i(x_i^*, x_{-i}^*) \geq u_i(x_i^{}x_{-i}^*) \qquad \forall i \in \mathcal{I}, x_i \in \Delta_i \textit{ s.t. } x_i \neq x_i^*$$

Nash equilibria based on pure strategies may not exist in some games. However, a Nash equilibrium based on mixed strategies is always available.

In Graph Transduction Game, a Nash equilibrium moves mixed strategies to extreme strategies resulting in globally consistent labelling of the dataset [38]. When equilibrium is found, data points class is represented by the strategy with the highest probability:

$$\phi_i = \arg\max_{h=1,\dots,c} x_{ih}$$

Computing the Nash equilibrium In this setup, an *evolutionary approach* is used to compute Nash equilibria. They can be found through a dynamical system belonging to the class of *Replicator Dynamics* [39] introduced in [41]. Within this context, the mixed strategies together compose a multi-population where only the fittest pure strategies are kept whereas the others go extinct.

In an evolutionary scenario, Nash equilibria can be seen as the evolution of the fittest strategies in a multi-population of strategies. Such an evolution is the result of playing a game repeatedly. The discrete version of the dynamics is the following:

$$x_{ih}^{(t+1)} = x_{ih}^{(t)} \frac{u_i(e_i^{(h)}, x_{-i}^{(t)})}{u_i(x^{(t)})}$$

where t defines the current iteration of the process. The exit condition of dynamics can be a maximum number of iteration to reach or when there is a non-significant difference between two consecutive steps.

2.2.2 Graph Transduction Games for VSD

GTG requires as input three components:

1. The set of labelled players L and the set of unlabeled players U.
2. The similarity graph between the players.
3. An initial assignment between players and labels.

2.2.2.1 Players definition

GTG framework relies on the definition of a graph in which each node is a player of the noncooperative game. In this specific VSD setting, each element in the set of nodes $V = L \cup U$ corresponds to a pair <image, verb>, whereas each available strategy corresponds to a possible sense of the verb. For labelled players L, the verb sense (the fittest pure strategy) is known, while for players in U, it has to be inferred (Figure 2.9).

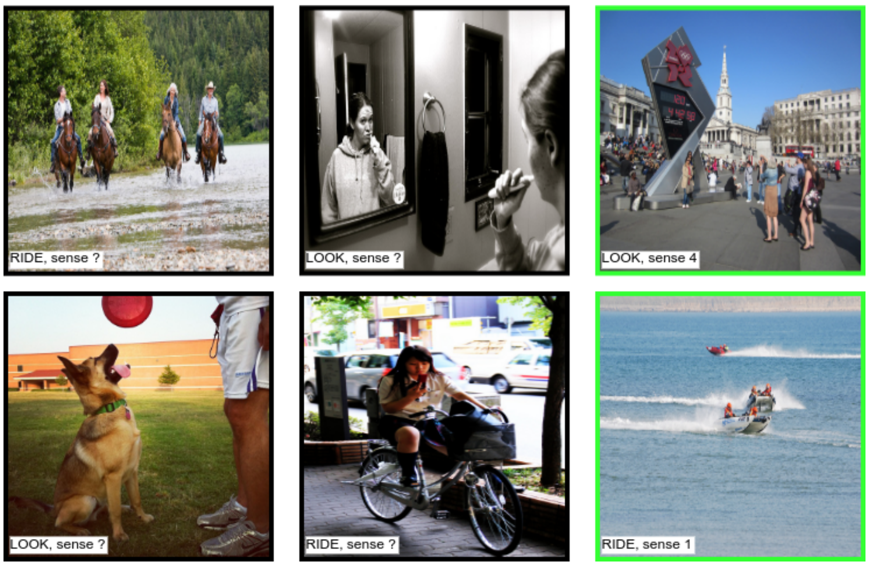


Figure 2.9: Set of labelled (green) and unlabeled (black) players.

2.2.2.2 Similarity between players

The similarity matrix ω is the adjacency matrix of the graph edges. It contains the similarity between each pair of players. The similarity function is the cosine, as defined in Section 2.1.3 and it is computed on the feature vectors (f_i) of the dataset images (Section 2.1.4):

$$A_{ij} = f_i \cdot f_j$$

The similarity between each pair of players i and j quantifies the effect that the player i has on player j, during the strategy selection. The more the players are similar the more likely they will share the same strategy (which means the same verb sense).

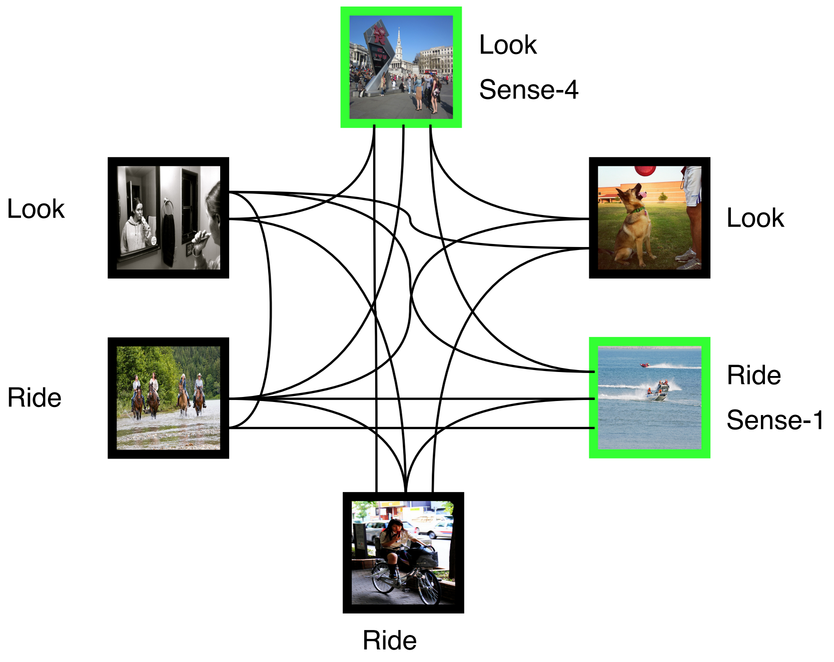


Figure 2.10: Set of labelled (green) and unlabeled (black) players.

2.2.2.3 Probability Initialisation

Strategies are initialised into a probability matrix x (in the evolutionary notation, at the step $x^{(0)}$) in which the rows are the players and the columns are the strategies. It is taken into account the distinction between *labelled* players L and *unlabeled* players U (where each player is a $< image, verb >$ pair). Players in L play only extreme strategies i.e. with a 1 probability on the strategy which matches their label, resulting in a one-hot vector.
For each player ($< image, verb >$ pair) in U the strategies are uniformly initialised among the feasible classes (senses that belong to the input verb), so just a submatrix is taken into account. Each cell x_{ih} is initialised as follows:

$$x_{ih} = \begin{cases} 1 & \text{if } i \text{ is labelled and have sense } h \\ \frac{1}{|S_i|} & \text{if } i \text{ is unlabeled and } h \in S_i \\ 0 & \text{otherwise} \end{cases}$$

where x_{ih} corresponds to the probability that the i-th player chooses strategy h while S_i is the set of possible senses associated with the verb paired with the image (pair i). An example to clarify this point is that if the target player has the verb *play*, the uniform probability is distributed only among the strategies which match the senses of *play*; the probabilities of all the other strategies are set to zero.

2.2.3 Algorithmic pipeline

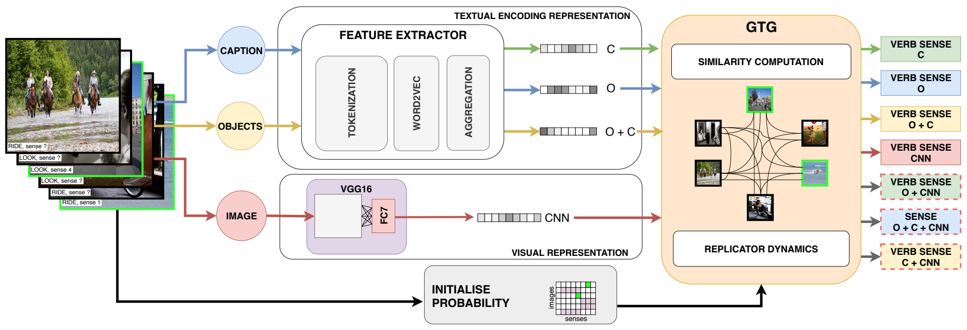


Figure 2.11: Pipeline of the algorithm considering both labelled (green border) and unlabeled images (black border).

The whole pipeline of the GTG Multimodal Verb Sense Disambiguation algorithm used on VerSe dataset is summarised in Figure 2.11 and is composed of the following steps:

1. At the beginning of the process, only a partition of the dataset is already labelled (at least an observation per class). Each observation is represented by a pair <image, verb> the label information (if known) is the verb sense identifier.

2. Each image i is encoded either in a unimodal (image/text/labels) or a multi-modal (concatenation of image/text/labels) way, as described in Section 2.1.4, resulting in $< f_i$, verb> pairs.

3. Each $< f_i$, verb> pair is used as a node in the graph which represents a player in the Graph Transduction Game. Graph nodes are connected through each others computing a pairwise similarity function between each data point, it represents graph weights.
   As in Equation 2.2, the similarity function is the cosine similarity.

4. Each row of the strategy space S is initialised with a uniform probability distribution among the senses of the input verb, each column represents a verb sense, hence a strategy in the GTG. Regarding the rows related to labelled data points, since their sense is already chosen, a 1 probability is set on their extreme strategy matching their class label.

5. Replicator Dynamics are run until a Nash equilibrium is found. In an algorithmic view this point is reached when one of the following conditions is satisfied:

    - the difference between the strategies in two successive steps is not significant, hence they remain unchanged.
    - Replicator Dynamics have reached a number of iterations in which according to empirical observations, they should have reached an equilibrium.

6. Once a Nash equilibrium is found, every player has selected a strategy which match its best option taking into account the strategies of its opponents. In classification terms this means that a consistent labelling of unlabeled data points is obtained.

Chapter 3

Experiments

In this section the configurations used to measure the performances of GTG Visual Sense Disambiguation are listed. Such results are then compared with the ones obtained in Gella et al. 2019.
All the unimodal representations of textual and visual data has been used, including both GOLD and PRED settings.

(Deep) Canonical Correlation Analysis The advanced multimodal representation techniques employed in Gella et al. 2019 to combine data, i.e. Canonical Correlation Analysis and its non-linear version, produced performances poorer than the simpler and fast vector concatenation. Even if such methods had interesting results in other researches [85] their usage is not worthy with VerSe. Just to replicate the experiments, in order to have a baseline with Gella et al. 2019 they needed additional data i.e. Flickr30k dataset and significant time to train the model. For this reason, they have not been taken into account in the GTG setup since their usage was not promising.

3.1 Experiments setup

In order to have consistent and comparable results, the experiments have been performed on the same dataset used in Gella et al. 2019 and the same performance quantifier has been used (accuracy metric). Considering the different features and their combinations, there are 7 possible setups for the experiments: captions (C), object labels (O), captions with object labels (C+O), CNN features (VIS), CNN features concatenated to captions (CNN+C), CNN features concatenated to object labels (CNN+O) and CNN features concatenated to captions with object labels (CNN+O+C). They go up to 13, due to the duplication of textual configurations of GOLD and PRED textual sources.

In Gella et al. 2019, once the encoding part is performed, the experiments are purely deterministic since they do not rely on stochastic parts, like samplings. In this semi-supervised setup, the labels are available for all the data, so we want to simulate the situation in which only a sample of VerSe is being labelled by randomly sampling labelled elements. The sampling is performed on n (s.t. $n \geq 1$) data points per class (verb sense). To have an unbiased evaluation which is not affected by the stochasticity of the sampling process, multiple seeds are taken. Each experiment is replicated on 15 different seeds, resulting in 15 *pilot runs*, in a way that each run has a different initial sample. Once the scores have been gathered they aggregated in the form: **mean $\pm$ standard deviation**.

In this setup, the chosen exit condition of the Replicator Dynamics is to reach a maximum of 10 iterations as suggested in [36].

3.1.1 Ablation studies

In some cases, a single labelled element per class may be sufficient to overcome the results of Gella et al. 2019 since the semi-supervised setup generally lies beyond the unsupervised one. However, an interesting experiment is to understand how the labelled sample size affects the accuracy of the classification. For this reason, the experiments with GTG have been carried out considering an increasing number of labelled elements per sense and accounting for the accuracy on the unlabeled elements. The number of labelled elements varies from 1 to 13, after that point the difference became non-significant.

The distribution of verb senses in the dataset is not uniform but is *Zipfian*, so there are many verbs which are related to a few data points and only a few verbs which are related to many data points. For this reason, a 1-label per sense increase will be much more influent in some classes rather than other ones. Due to this sense distribution, on sample size increase some classes will label all their available points very fast.

To keep having unlabeled data points for each class, the following label sampling rule is used:

$$g(n) = \begin{cases} n & \text{if } n < m \\ m - 1 & \text{otherwise} \end{cases}$$

where n is the number of data points for each class that is needed for the experiment and m is the total number of the data points of the class.

3.1.2 Modern image classifiers

The disambiguation method used in Gella et al. 2019 has been designed around 2015. As described in Section 2.1.4.1, at that moment the best image classifier available was VGGNet. In the next years, better networks have been released like *Inception* and *ResNet*. Nevertheless, these networks have been trained with the goal of classifying images.

An interesting experiment to perform is in this GTG setup, is to employ a state-of-the-art object detector for PRED object label generation: *Faster R-CNN* [29]. This tool has been designed to provide the actual content of the image rather than an estimate of the possible classes of an image. The idea is that an object detector should be better for pictures with several objects in it. A difference between an image classifier like VGGNet and an object detector like Faster R-CNN in terms of produced labels is that the latter might generate non-unique object labels, to have comparable results the duplicates have been dropped.

Moreover, a modern CNN might affect also the textual data, because the captions that are generated in the PRED setup are obtained with a neural network trained on a VGGNet. Thus, another experiment has been performed in which the PRED image caption tool has been replaced with a more recent one [30], which is based on a *ResNet101* structure rather than a VGGNet.

Chapter 4

Results

In Table 4.1 are reported the results of Gella et al. 2019 (first row) compared with the game-transductive approach using several labelled sample sizes; using the GOLD configuration for textual experiments. An explanation about specific encoding abbreviation can be found in Section 3.1. The table content is measured using the accuracy metric.

GOLD	Verb type	Textual			VIS (CNN)	Concat (CNN+)		
		O	C	Combined (O+C)		Object (O)	Captions (C)	Combined (O+C)
Paper results	Motion	54.60	73.30	75.60	58.30	66.60	74.70	73.80
PAMI 2019	Non-Motion	57.00	72.70	72.60	56.10	66.00	72.20	71.30
1 label	Motion	73.32 ± 4.4	73.36 ± 6.0	74.15 ± 5.5	73.33 ± 5.9	74.68 ± 3.4	74.59 ± 5.3	74.07 ± 5.5
	Non Motion	71.68 ± 4.6	71.35 ± 4.4	75.78 ± 4.0	64.39 ± 6.6	71.43 ± 4.9	70.81 ± 4.7	74.80 ± 4.0
2 labels	Motion	79.38 ± 5.2	83.06 ± 2.7	83.27 ± 2.6	78.83 ± 4.6	80.72 ± 4.5	83.98 ± 2.9	83.56 ± 3.2
	Non Motion	82.49 ± 3.2	81.75 ± 2.9	82.24 ± 4.0	80.80 ± 4.0	83.41 ± 2.8	81.78 ± 2.0	81.72 ± 3.0
3 labels	Motion	80.95 ± 4.9	82.02 ± 2.5	84.30 ± 3.0	81.44 ± 4.3	81.13 ± 5.2	82.04 ± 3.2	84.64 ± 3.4
	Non Motion	86.11 ± 2.3	87.60 ± 2.1	87.69 ± 1.9	87.60 ± 2.2	86.24 ± 2.4	87.89 ± 1.4	87.80 ± 1.8
4 labels	Motion	81.85 ± 4.3	81.33 ± 4.4	85.17 ± 3.6	83.80 ± 4.5	82.98 ± 4.2	83.24 ± 3.3	84.84 ± 4.2
	Non Motion	86.56 ± 2.4	87.60 ± 2.2	88.05 ± 2.0	87.57 ± 2.4	87.18 ± 2.3	87.61 ± 2.2	87.97 ± 2.2
5 labels	Motion	82.68 ± 3.6	83.40 ± 3.1	86.70 ± 4.1	85.57 ± 4.1	83.36 ± 3.4	82.98 ± 3.0	87.45 ± 3.2
	Non Motion	86.84 ± 2.4	88.64 ± 1.8	88.63 ± 2.2	87.62 ± 2.3	86.80 ± 2.2	88.68 ± 2.5	88.87 ± 2.4
6 labels	Motion	83.42 ± 3.3	84.00 ± 3.5	87.15 ± 3.9	86.83 ± 3.3	85.10 ± 3.0	84.39 ± 3.5	87.70 ± 3.9
	Non Motion	86.87 ± 1.7	89.07 ± 1.9	88.97 ± 1.4	88.40 ± 3.0	87.09 ± 2.3	89.57 ± 1.4	89.10 ± 1.7
7 labels	Motion	83.81 ± 3.3	88.59 ± 2.0	88.85 ± 2.8	91.60 ± 1.5	85.03 ± 2.6	88.74 ± 1.9	89.58 ± 2.1
	Non Motion	89.44 ± 2.4	92.21 ± 2.0	92.20 ± 2.0	90.07 ± 2.8	89.55 ± 2.2	92.81 ± 2.0	93.01 ± 1.9
8 labels	Motion	84.12 ± 3.6	88.89 ± 1.9	89.19 ± 2.0	91.97 ± 1.5	86.41 ± 2.8	89.44 ± 0.9	89.34 ± 2.0
	Non Motion	89.59 ± 2.7	92.59 ± 2.0	92.42 ± 1.6	90.76 ± 3.0	89.40 ± 2.7	93.65 ± 1.6	92.39 ± 2.3
9 labels	Motion	87.45 ± 1.1	89.58 ± 0.7	90.00 ± 0.6	92.44 ± 1.6	88.46 ± 2.3	89.81 ± 0.7	90.18 ± 0.6
	Non Motion	92.24 ± 2.1	93.36 ± 1.8	92.53 ± 1.7	93.47 ± 2.8	92.25 ± 2.2	93.97 ± 2.0	93.22 ± 1.6
10 labels	Motion	87.51 ± 1.1	89.90 ± 0.6	90.28 ± 0.4	92.51 ± 1.6	88.55 ± 2.0	90.15 ± 0.5	90.40 ± 0.5
	Non Motion	92.99 ± 2.0	93.20 ± 2.1	92.93 ± 1.7	94.28 ± 1.7	93.37 ± 1.1	93.61 ± 2.1	93.69 ± 1.5
11 labels	Motion	87.76 ± 1.3	90.36 ± 0.7	90.49 ± 0.4	93.16 ± 0.9	89.04 ± 2.3	90.64 ± 0.7	90.56 ± 0.4
	Non Motion	93.15 ± 1.5	93.82 ± 1.5	93.24 ± 1.1	94.51 ± 1.7	93.26 ± 0.9	93.98 ± 1.7	93.41 ± 1.2
12 labels	Motion	88.85 ± 1.6	91.37 ± 0.4	91.53 ± 0.4	94.21 ± 1.0	90.20 ± 2.7	91.56 ± 0.3	91.60 ± 0.4
	Non Motion	93.29 ± 0.9	93.88 ± 1.7	93.19 ± 1.2	94.43 ± 1.7	93.22 ± 0.9	93.75 ± 1.6	93.19 ± 1.2
13 labels	Motion	90.33 ± 2.3	91.67 ± 0.2	91.85 ± 0.1	94.51 ± 0.9	91.65 ± 3.1	91.82 ± 0.1	91.89 ± 0.1
	Non Motion	93.42 ± 1.3	93.55 ± 1.5	93.10 ± 1.2	94.65 ± 1.8	93.44 ± 1.3	93.92 ± 1.8	93.13 ± 1.2

Table 4.1: Results of Gella et al. 2019 *vs.* GTG for GOLD setting

These results consist in the means of all the 15 runs coupled with their standard deviations. In Gella et al. 2019, the results have been split according to Levin

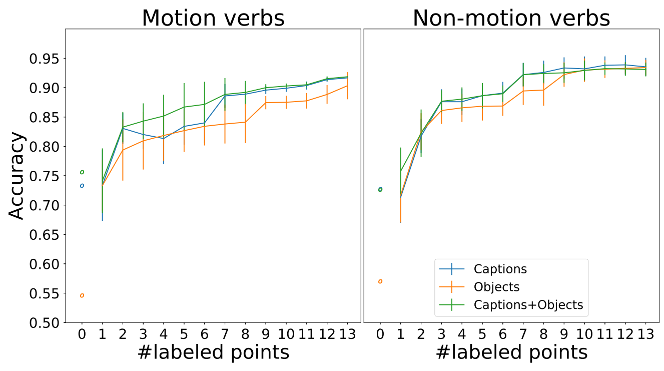
PRED	Verb type	Textual			VIS (CNN)	Concat (CNN+)		
		O	C	Combined (O+C)		Object (O)	Captions (C)	Combined (O+C)
Paper results PAMI 2019	Motion	65.1	54.9	61.6	58.3	72.6	63.6	66.5
	Non-Motion	59.0	64.3	65.0	56.1	63.8	66.3	66.1
1 label	Motion	71.17 ± 6.4	71.52 ± 3.9	71.87 ± 5.1	73.33 ± 5.9	72.97 ± 6.3	73.83 ± 4.3	74.01 ± 3.6
	Non Motion	64.37 ± 5.2	72.18 ± 5.4	73.34 ± 4.4	64.39 ± 6.6	65.58 ± 6.0	73.31 ± 4.9	73.31 ± 4.6
2 labels	Motion	79.54 ± 4.9	77.09 ± 4.3	78.04 ± 5.0	78.83 ± 4.6	80.17 ± 4.4	77.70 ± 3.6	78.27 ± 3.7
	Non Motion	75.32 ± 4.1	84.34 ± 3.3	82.71 ± 3.7	80.80 ± 4.0	77.34 ± 3.4	84.31 ± 3.7	83.07 ± 3.7
3 labels	Motion	80.09 ± 4.3	79.76 ± 3.6	80.00 ± 3.7	81.44 ± 4.3	80.14 ± 4.4	79.16 ± 3.6	80.05 ± 4.0
	Non Motion	82.95 ± 3.3	89.01 ± 1.5	88.81 ± 1.9	87.60 ± 2.2	85.63 ± 3.7	89.17 ± 1.9	88.60 ± 2.4
4 labels	Motion	79.29 ± 4.7	82.16 ± 3.5	82.20 ± 3.6	83.80 ± 4.5	79.88 ± 4.4	81.95 ± 3.6	82.22 ± 4.1
	Non Motion	85.12 ± 4.4	89.49 ± 1.6	89.20 ± 2.0	87.57 ± 2.4	87.31 ± 2.9	89.47 ± 1.5	89.05 ± 1.8
5 labels	Motion	81.23 ± 3.8	85.21 ± 3.2	84.11 ± 3.0	85.57 ± 4.1	82.43 ± 4.1	85.04 ± 3.2	84.67 ± 3.6
	Non Motion	87.40 ± 3.1	90.27 ± 2.4	89.93 ± 2.3	87.62 ± 2.3	87.90 ± 2.6	90.28 ± 2.2	89.94 ± 2.3
6 labels	Motion	82.39 ± 4.8	86.91 ± 2.6	86.72 ± 2.5	86.83 ± 3.3	83.78 ± 4.8	87.28 ± 2.7	87.46 ± 2.3
	Non Motion	87.86 ± 3.1	90.49 ± 2.0	90.04 ± 2.0	88.40 ± 3.0	88.69 ± 3.3	90.45 ± 2.2	89.81 ± 2.4
7 labels	Motion	83.78 ± 5.2	88.05 ± 1.1	87.99 ± 1.0	91.60 ± 1.5	84.78 ± 3.7	88.47 ± 1.1	88.42 ± 1.0
	Non Motion	88.94 ± 2.9	92.33 ± 1.6	91.61 ± 1.5	90.07 ± 2.8	90.56 ± 2.7	92.22 ± 1.7	91.41 ± 2.6
8 labels	Motion	85.71 ± 4.2	88.32 ± 0.7	88.18 ± 0.7	91.97 ± 1.5	86.49 ± 3.1	88.65 ± 1.0	89.01 ± 1.2
	Non Motion	89.51 ± 3.5	92.87 ± 1.5	92.43 ± 2.0	90.76 ± 3.0	89.65 ± 2.6	93.45 ± 1.5	93.23 ± 2.2
9 labels	Motion	87.90 ± 2.9	88.53 ± 0.8	88.62 ± 0.8	92.44 ± 1.6	88.54 ± 2.5	88.95 ± 0.9	89.24 ± 1.2
	Non Motion	91.39 ± 2.7	92.95 ± 2.0	92.65 ± 2.1	93.47 ± 2.8	91.99 ± 2.6	93.30 ± 2.2	93.11 ± 2.1
10 labels	Motion	88.88 ± 2.7	88.93 ± 0.6	89.26 ± 0.7	92.51 ± 1.6	88.86 ± 2.7	89.47 ± 0.7	89.58 ± 0.6
	Non Motion	92.15 ± 3.1	93.39 ± 2.0	92.56 ± 2.3	94.28 ± 1.7	93.95 ± 1.6	94.03 ± 1.6	93.57 ± 2.2
11 labels	Motion	89.70 ± 2.6	89.45 ± 0.5	89.63 ± 0.5	93.16 ± 0.9	90.07 ± 1.9	90.00 ± 0.6	89.89 ± 0.7
	Non Motion	94.00 ± 1.5	94.44 ± 1.8	93.99 ± 1.7	94.51 ± 1.7	94.21 ± 1.6	94.46 ± 1.8	93.77 ± 1.6
12 labels	Motion	90.44 ± 2.6	90.94 ± 0.7	91.07 ± 0.6	94.21 ± 1.0	90.70 ± 2.0	91.38 ± 0.7	91.29 ± 0.9
	Non Motion	93.84 ± 1.6	94.43 ± 1.9	93.89 ± 1.7	94.43 ± 1.7	94.02 ± 1.6	94.12 ± 1.7	93.94 ± 1.7
13 labels	Motion	91.35 ± 2.2	91.49 ± 0.7	91.63 ± 0.6	94.51 ± 0.9	91.14 ± 1.7	91.77 ± 0.4	91.77 ± 0.7
	Non Motion	94.19 ± 1.7	94.01 ± 1.8	93.69 ± 1.7	94.65 ± 1.8	94.27 ± 1.8	94.17 ± 1.8	94.31 ± 1.8

Table 4.2: Results of Gella et al. 2019 *vs.* GTG for PRED setting

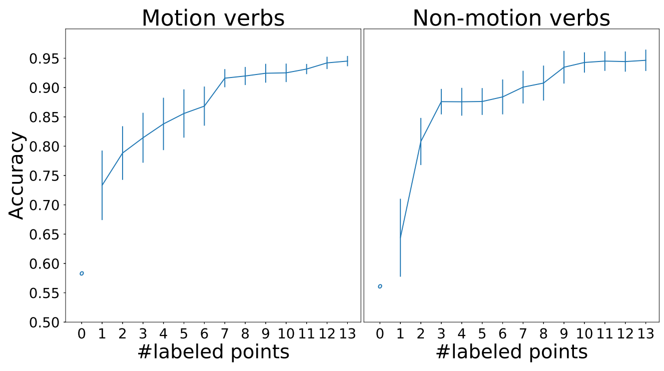
semantic verb classes (motion and non-motion verbs) [18] since considering them jointly in some cases would have affected the final score. Moreover, the verbs related to the images which compose the dataset are almost uniformly distributed among these two classes i.e. 1812 for motion verbs and 1698 for non-motion verbs.

In the table, it is shown how the labelled sample size affects the accuracy of the algorithm. The same tabular organisation has been used for Table 4.2 for PRED configuration for textual experiments.
The comparison between the two approaches can be seen also from the plots in Figure 4.1 and Figure 4.2, in which the performances are plotted varying the number of labelled data points per class in the vertical axes and the disambiguation accuracy in the horizontal axes. The circles in the plots at x-value zero, represent the scores obtained in Gella et al. 2019 which are used as the baseline. As can be seen, the performances with 1 labelled point per sense are comparable with Gella et al. 2019, this means that a dynamic classification, performed after a convergence point can improve the performance. Adding more labelled information i.e. 2 or more labelled data points per class, dramatically outperforms the baseline. Theoretically the more the labelled information increase, the more the accuracy means will converge, reducing at each step the standard deviations.

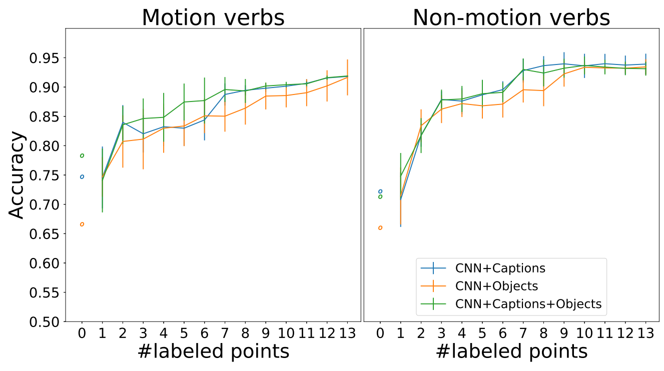
In GOLD setting (Fig.4.1), for the experiments performed using unimodal representations of object labels (O) and visual features (CNN); the GTG approach



(a) text



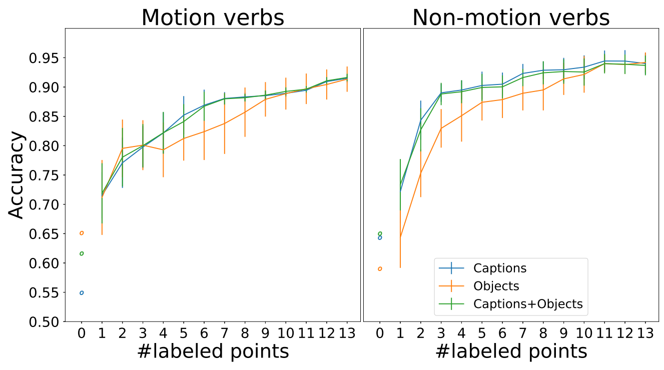
(b) cnn



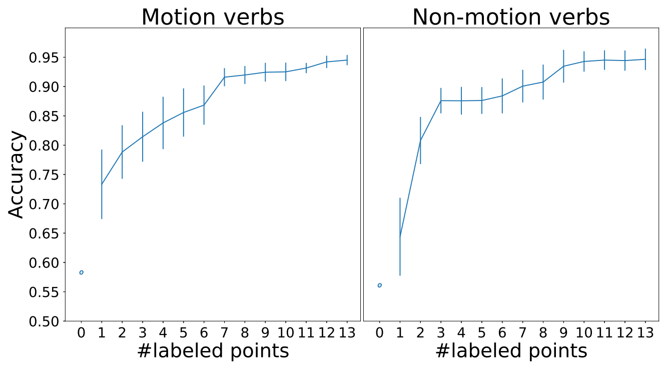
(c) cnn+text

Figure 4.1: GOLD results for text data, cnn and cnn+text varying the number of labelled points in comparison with Gella et al. approach (circles).

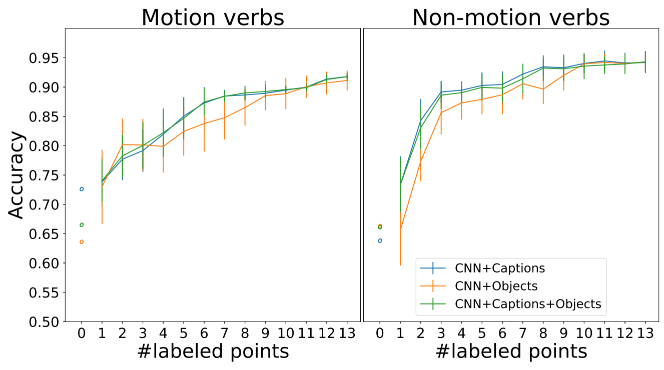
outperforms Gella et al. accuracies needing just 1 labelled point per class. Whereas for caption data, they are quite aligned; especially if taking into account standard deviations. This has an impact on all derived configuration, both unimodal (cap-



(a) text



(b) cnn



(c) cnn+text

Figure 4.2: PRED results for text data, cnn and cnn+text varying the number of labelled points in comparison with Gella et al. approach (circles).

tions+object labels) and multimodal. In fact, in such cases, the GTG score is aligned with Gella et al. 2019. In PRED textual configuration (Fig. 4.2), all GTG experiments outperform Gella et al. 2019 results when two labelled points are considered.

When only 1 labelled point is used, the performances of Gella et al. 2019 are in the range of variances of GTG. In general, it can be noted that as expected, the higher the number of labelled points per sense, the higher the overall accuracy and smaller the standard deviation. The accuracy curve follows a logarithmic growth i.e. the variation of the number of labels has a relevant role when they are few, whereas, with more than 6–7 labelled points per class, the accuracy starts converging.

4.1 Performances with modern DNN

The scores obtained replacing Faster R-CNN as label extractor and ResNet101 as base for the caption generator are reported in Table 4.3. In some cases, both the mean accuracy and the overall standard deviation have been reduced resulting in a more stable score. On the other hand, in some other cases, both the mean and the standard deviation increased. Thus, the scores are aligned to the ones obtained with standard PRED.

ModernPRED	Verb type	Textual			VIS (CNN)	Concat (CNN+)		
		O	C	Combined (O+C)		Object (O)	Captions (C)	Combined (O+C)
Paper results	Motion	65.1	54.9	61.6	58.3	72.6	63.6	66.5
PAMI 2019	Non-Motion	59.0	64.3	65.0	56.1	63.8	66.3	66.1
1 label	Motion	70.55 ± 5.1	70.92 ± 5.0	71.98 ± 4.6	73.33 ± 5.9	72.22 ± 4.4	74.52 ± 4.0	73.78 ± 4.2
	Non Motion	71.92 ± 3.6	70.19 ± 4.1	71.17 ± 3.3	64.39 ± 6.6	71.53 ± 4.4	70.55 ± 3.8	71.12 ± 3.5
2 label	Motion	73.63 ± 4.4	76.19 ± 4.8	76.62 ± 5.7	78.83 ± 4.6	74.68 ± 5.0	77.44 ± 4.6	77.82 ± 4.9
	Non Motion	83.94 ± 3.6	80.58 ± 3.9	81.98 ± 3.0	80.80 ± 4.0	83.10 ± 2.9	80.32 ± 4.1	81.77 ± 2.9
3 label	Motion	74.55 ± 5.3	77.01 ± 2.9	77.67 ± 2.9	81.44 ± 4.3	76.45 ± 5.3	78.33 ± 2.3	78.66 ± 2.7
	Non Motion	87.70 ± 2.4	87.27 ± 2.8	88.06 ± 2.6	87.60 ± 2.2	87.00 ± 2.3	87.07 ± 3.1	87.30 ± 3.1
4 label	Motion	76.45 ± 3.5	77.67 ± 3.2	77.81 ± 3.0	83.80 ± 4.5	78.15 ± 5.1	79.47 ± 3.9	79.67 ± 3.6
	Non Motion	88.50 ± 2.4	86.71 ± 2.3	88.17 ± 1.6	87.57 ± 2.4	87.97 ± 2.8	87.13 ± 2.2	87.94 ± 2.0
5 label	Motion	76.52 ± 4.2	79.39 ± 3.8	79.31 ± 3.9	85.57 ± 4.1	78.86 ± 6.0	80.44 ± 4.0	80.43 ± 4.3
	Non Motion	89.57 ± 2.6	87.60 ± 2.0	89.57 ± 1.9	87.62 ± 2.3	88.78 ± 2.9	88.17 ± 2.8	89.17 ± 2.2
6 label	Motion	76.97 ± 3.3	80.05 ± 4.1	79.25 ± 4.1	86.83 ± 3.3	79.49 ± 6.2	81.30 ± 3.9	81.31 ± 3.9
	Non Motion	89.88 ± 2.8	88.08 ± 2.3	89.07 ± 2.6	88.40 ± 3.0	89.42 ± 2.7	88.57 ± 3.0	90.13 ± 3.1
7 label	Motion	79.73 ± 4.3	85.96 ± 3.6	86.03 ± 3.2	91.60 ± 1.5	82.10 ± 5.1	87.92 ± 2.4	88.19 ± 2.7
	Non Motion	92.57 ± 3.0	90.41 ± 2.4	91.50 ± 2.4	90.07 ± 2.8	91.66 ± 3.0	90.68 ± 2.6	91.66 ± 3.1
8 label	Motion	78.86 ± 4.3	87.39 ± 2.8	85.89 ± 3.3	91.97 ± 1.5	83.25 ± 4.4	88.56 ± 2.2	88.49 ± 2.6
	Non Motion	91.78 ± 3.0	90.78 ± 2.3	92.10 ± 2.4	90.76 ± 3.0	91.09 ± 3.0	91.40 ± 2.4	92.77 ± 3.3
9 label	Motion	86.22 ± 2.2	88.72 ± 0.8	88.71 ± 1.0	92.44 ± 1.6	87.44 ± 1.8	89.71 ± 0.9	89.75 ± 1.1
	Non Motion	94.79 ± 1.6	91.25 ± 1.5	92.68 ± 2.3	93.47 ± 2.8	94.71 ± 1.6	92.03 ± 1.8	93.48 ± 1.9
10 label	Motion	87.19 ± 2.5	89.13 ± 0.6	88.70 ± 1.0	92.51 ± 1.6	88.11 ± 1.3	89.56 ± 0.9	89.73 ± 0.8
	Non Motion	95.05 ± 1.6	91.49 ± 1.9	92.70 ± 2.4	94.28 ± 1.7	95.06 ± 1.6	92.37 ± 1.9	93.15 ± 2.2
11 label	Motion	87.68 ± 2.4	89.50 ± 0.5	89.30 ± 1.0	93.16 ± 0.9	88.79 ± 1.4	90.18 ± 0.7	90.27 ± 0.9
	Non Motion	94.90 ± 1.8	93.30 ± 1.2	94.13 ± 1.7	94.51 ± 1.7	94.90 ± 1.7	93.53 ± 1.4	94.00 ± 1.7
12 label	Motion	89.02 ± 2.1	91.09 ± 0.5	90.80 ± 0.8	94.21 ± 1.0	90.60 ± 1.1	91.72 ± 0.5	91.81 ± 0.5
	Non Motion	94.95 ± 1.8	93.22 ± 1.3	94.64 ± 1.8	94.43 ± 1.7	94.89 ± 1.8	93.22 ± 1.3	94.46 ± 1.8
13 label	Motion	89.45 ± 2.2	91.34 ± 0.5	91.14 ± 0.5	94.51 ± 0.9	91.47 ± 0.7	91.79 ± 0.3	91.90 ± 0.3
	Non Motion	95.39 ± 1.7	93.42 ± 1.5	95.03 ± 1.8	94.65 ± 1.8	94.92 ± 1.8	93.49 ± 1.5	94.67 ± 1.9

Table 4.3: Results considering Faster R-CNN and a modern DNN for PRED textual setting.

this means that VGGNet is still a worth feature extractor with respect to ResNet. The use of an object detector did not affect the label extraction process much.

4.2 Baseline heuristics

In Gella et al 2019, to evaluate the disambiguation algorithm, their method has been compared with two typical NLP baseline heuristics:

- **First Sense heuristics (FS):** For each verb to disambiguate, the first verb sense listed in a knowledge-base is returned. In this case such information is retrieved from OntoNotes. This approach aims to use information which is independent of the current data. Exploiting a semantic network; Usually, senses are listed by frequency. Such an approach for VerSe dataset performed an accuracy of 70.8% for motion verbs and an accuracy of 80.6% for non-motion verbs.
  With respect to the transductive approach, such baseline is overcome only with one label per class for motion verbs and two labels per class for non-motion verbs. However such statement is valid only if standard deviations are dropped. When they are taken into account, to be sure that such baseline is outperformed it is necessary to have 3 labels for motion verbs and 3–4 labels for non-motion verbs especially if PRED setup is considered.

- **Most Frequent Sense heuristics (MFS):** For each verb to be disambiguated, the most frequent sense of that verb appearing in dataset ground truth is used. This technique requires full knowledge of dataset labels; in fact, it is considered as a supervised heuristics. Such an approach is quite similar to FS heuristics, but in this case, the evaluation is biased by the dataset, being dependent on it. This approach applied to VerSe dataset produced an accuracy of 86.2% for motion verbs and an accuracy of 90.7% for non-motion verbs. In this case, the relatedness of data points from the same dataset affected a lot the sense assignment. This means which is likely that in VerSe the sense distributions tend towards senses which are not the most common.
  This heuristics is quite hard to be overcome by the transductive approach. In fact, for motion verbs 7–8 labels per class are needed whereas, for non-motion verbs, 9 labels might not be enough if standard deviations are considered.
  Nevertheless as said at the beginning, MFS is considered as a supervised learning approach, which theoretically should always act as an upper bound for a semi-supervised approach, which should lie between its unsupervised and supervised counterparts.

Chapter 5

Conclusions & Future Work

In this thesis, the suitability of an evolutionary game-theoretic model has been shown in the task of multimodal verb sense disambiguation. A first overview of the well-known word sense disambiguation task has been given. Then, it has been shown how it has been transferred to multimodal data; circumscribing the task just to verbs. After that, all the essential information about the reference paper has been explored. Passing from the goals, dataset, the encodings and the algorithm. Hence, the Graph Transduction Game algorithm has been introduced with a brief explanation on evolutionary game theory fundamentals. Eventually, experimental setups and their respective results have been described.

In this chapter, the main advantages and disadvantages of the discussed approach will be resumed and discussed. After that, some possible future developments which may enhance the work are listed.

Advantages and disadvantages of the method One of the main advantages of the transductive method is that verb senses have been converted from entities to labels. In Gella et al. 2019, since the learning approach is unsupervised and founded on Lesk algorithm [15], there is the need to have dictionary definitions and images crawled from the web and encode them to entities which represent verb senses.
Upon initial analysis, this implies a big effort in term of data gathering. Although, once this process has been performed, the disambiguation part is quite simple. However, an important fact is that the quality of such data can affect a lot the final result. In fact, in a textual scenario, the dictionary examples could lead towards a very specific context of usage of the verb, resulting in a biased representation. Whereas, for the visual scenario, the web engine might retrieve wrong images, resulting in data outliers which act as noise for the representation (which in certain situations may also be worthy). All these incoveniences, are going to be avoided by treating senses only as labels in the transductive approach. This means that the learning approach moves from the fully unsupervised to the semi-unsupervised one. The transductive approach has been selected for VerSe because it has already been used in WSD. It has been applied in a fully textual scenario in a former research [24]

scoring good results. Nevertheless, in such a paper, the payoffs depend also on the similarity between senses of words appearing nearby the target word. That is contextual information which is not available in this setup.

The power of the approach proposed in this thesis relies on the transductive nature of the Graph Transduction Games which performs the inference for the sense of a verb using not only on the similarity with the labeled samples but considering also the unlabeled ones, leading to a more precise boundaries definition. The similarity is measured across multimodal image representations which come from the same dataset and not on sense-representing entities which come from knowledge bases. This means that since they come from the same dataset, it is more likely that similar elements belong to the same class. Whereas, with sense-data fetched with a black box crawling approach (as in Gella et al. 2019), this is not valid.

A first disadvantage of the transductive approach is the computational time. Starting both algorithms in equal conditions in which all the data has been aggregated and encoded, a much more fast evaluation will be performed in Gella et al. 2019 approach. Such an algorithm only has to compute the dot product between each image representation and sense representation and then extract the position of maxima. In GTG sense entities are not present, however, Replicator Dynamics is a loop in which matrix multiplication is performed and a convergence point has to be reached.

Another disadvantage of the transductive process is that this approach relies a lot on propagate information exploiting information coming from neighbouring observations, rather than learning to classify objects as in generative and discriminative models. This can result in a lack of generalization. However, it can be easily overcome through the training of a supervised classifier with the newly labeled dataset (as explored in [25]).

Future works: Prior knowledge In Section 2.2.2.3 the initialisation of the probability space initialisation has been described as a uniform distribution among the senses of the target verb. Such initialisation is the simplest and is good when no external information about classes is available. However, other probabilities distribution can be used. For instance in [24], a geometric distribution has been exploited to rank senses using prior knowledge from labeled data. Another initialisation technique using geometric distribution could be based on First Sense heuristic extracted from a knowledge base. It may work quite well due to the skewed nature of the verb senses.

An interesting solution which could be implemented relying on the setup discussed in this thesis consists of the usage of encoded sense entities from Gella et al. 2019. Since the cosine similarities between image representations and sense representation already lie between 0 and 1. They can be converted to probabilities in a way that their sum adds up to one. After that, standard Replicator Dynamics can be run. In this thesis, such experiments could not be performed due to the hard replicability of sense encoding performed in Gella et al. 2019.

Bibliography

[1] S. Gella, F. Keller, and M. Lapata, "Disambiguating visual verbs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 311–322, Feb 2019.

[2] S. Gella, M. Lapata, and F. Keller, "Unsupervised visual sense disambiguation for verbs using multimodal embeddings," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (San Diego, California), pp. 182–192, Association for Computational Linguistics, June 2016.

[3] S. Vascon, S. Aslan, A. Torcinovich, T. van Laarhoven, E. Marchiori, and M. Pelillo, "Unsupervised domain adaptation using graph transduction games," 2019.

[4] D. Yuan, J. Richardson, R. Doherty, C. Evans, and E. Altendorf, "Semi-supervised word sense disambiguation with neural models," in *COLING 2016*, 2016.

[5] Z. Zhong and H. T. Ng, "It makes sense: A wide-coverage word sense disambiguation system for free text," in *Proceedings of the ACL 2010 System Demonstrations*, (Uppsala, Sweden), pp. 78–83, Association for Computational Linguistics, July 2010.

[6] N. Loeff, C. O. Alm, and D. A. Forsyth, "Discriminating image senses by clustering with multimodal features," in *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, (Sydney, Australia), pp. 547–554, Association for Computational Linguistics, July 2006.

[7] K. Saenko and T. Darrell, "Unsupervised learning of visual sense models for polysemous words," in *Advances in Neural Information Processing Systems 21* (D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds.), pp. 1393–1400, Curran Associates, Inc., 2009.

[8] K. Barnard and M. Johnson, "Word sense disambiguation with pictures," *Artificial Intelligence*, vol. 167, no. 1, pp. 13 – 30, 2005. Connecting Language to the World.

[9] E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel, "Ontonotes: The 90% solution," in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, (Stroudsburg, PA, USA), pp. 57–60, Association for Computational Linguistics, 2006.

[10] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: An On-line Lexical Database*," *International Journal of Lexicography*, vol. 3, pp. 235–244, 12 1990.

[11] R. Navigli and S. P. Ponzetto, "BabelNet: Building a very large multilingual semantic network," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, (Uppsala, Sweden), pp. 216–225, Association for Computational Linguistics, July 2010.

[12] D.-T. Le, J. Uijlings, and R. Bernardi, "TUHOI: Trento universal human object interaction dataset," in *Proceedings of the Third Workshop on Vision and Language*, (Dublin, Ireland), pp. 17–24, Dublin City University and the Association for Computational Linguistics, Aug. 2014.

[13] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *ArXiv*, vol. abs/1504.00325, 2015.

[14] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *TACL*, vol. 2, pp. 67–78, 2014.

[15] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone," in *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, (New York, NY, USA), pp. 24–26, ACM, 1986.

[16] A. Kilgarriff and J. Rosenzweig, "Framework and results for english senseval," *Computers and the Humanities*, vol. 34, pp. 15–48, 04 2000.

[17] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009.

[18] B. Levin, *English verb classes and alternations: A preliminary investigation*. University of Chicago Press, 1993.

[19] Y. Berzak, A. Barbu, D. Harari, B. Katz, and S. Ullman, "Do you see what I mean? visual resolution of linguistic ambiguities," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 1477–1487, Association for Computational Linguistics, Sept. 2015.

[20] T. Botschen, I. Gurevych, J.-C. Klie, H. Mousselly-Sergieh, and S. Roth, "Multimodal frame identification with multilingual evaluation," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 1481–1491, Association for Computational Linguistics, June 2018.

[21] A. Erdem and M. Pelillo, "Graph transduction as a noncooperative game," *Neural Comput.*, vol. 24, pp. 700–723, Mar. 2012.

[22] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," tech. rep., Center for Automated Learning and Discovery, CMU, 2002.

[23] R. Tripodi, M. Pelillo, and R. Delmonte, "An evolutionary game theoretic approach to word sense disambiguation," *Proceedings of Natural Language Processing and Cognitive Science*, vol. 2014, pp. 39–48, 2015.

[24] R. Tripodi and M. Pelillo, "A game-theoretic approach to word sense disambiguation," *Comput. Linguist.*, vol. 43, pp. 31–70, Apr. 2017.

[25] I. Elezi, A. Torcinovich, S. Vascon, and M. Pelillo, "Transductive label augmentation for improved deep network learning," in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 1432–1437, IEEE, 2018.

[26] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. `http://is.muni.cz/publication/884893/en`.

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[28] J. Yang, J. Lu, D. Batra, and D. Parikh, "A faster pytorch implementation of faster r-cnn," *https://github.com/jwyang/faster-rcnn.pytorch*, 2017.

[29] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[30] R. Luo, B. Price, S. Cohen, and G. Shakhnarovich, "Discriminability objective for training descriptive captions," *arXiv preprint arXiv:1803.04376*, 2018.

[31] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156–3164, 2015.

[32] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

[33] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Conference Track Proceedings*, 2013.

[34] S. Vascon, M. Frasca, R. Tripodi, G. Valentini, and M. Pelillo, “Protein function prediction as a graph-transduction game,” *Pattern Recognition Letters*, 2018.

[35] S. Vascon, S. Aslan, A. Torcinovich, T. van Laarhoven, E. Marchiori, and M. Pelillo, “Unsupervised domain adaptation using graph transduction games,” in -, 2019.

[36] I. Elezi, A. Torcinovich, S. Vascon, and M. Pelillo, “Transductive label augmentation for improved deep network learning,” in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 1432–1437, Aug 2018.

[37] J. Nash, “Non-cooperative games,” *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[38] D. A. Miller and S. W. Zucker, “Copositive-plus lemke algorithm solves polymatrix games,” *Oper. Res. Lett.*, vol. 10, pp. 285–290, July 1991.

[39] J. Maynard Smith, *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[40] J. T. Howson, “Equilibria of polymatrix games,” *Management Science*, vol. 18, no. 5-part-1, pp. 312–318, 1972.

[41] J. W. Weibull, *Evolutionary game theory*. MIT press, 1997.

[42] J. K. Pritchard, M. Stephens, and P. Donnelly, “Inference of population structure using multilocus genotype data,” *Genetics*, vol. 155, no. 2, pp. 945–959, 2000.

[43] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[44] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 09 2014.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.

[46] W. Weaver, “Translation,” *Machine translation of languages*, vol. 14, pp. 15–23, 1955.

[47] Y. Wilks and M. Stevenson, “The Grammar of Sense: Is word-sense tagging much more than part-of-speech tagging?,” *arXiv e-prints*, pp. cmp–lg/9607028, Jul 1996.

[48] N. Ide and J. Véronis, “Introduction to the special issue on word sense disambiguation: The state of the art,” *Comput. Linguist.*, vol. 24, pp. 2–40, Mar. 1998.

[49] R. Navigli, “Word sense disambiguation: A survey,” *ACM Comput. Surv.*, vol. 41, pp. 10:1–10:69, Feb. 2009.

[50] R. Navigli, “A quick tour of word sense disambiguation, induction and related approaches,” in *SOFSEM 2012: Theory and Practice of Computer Science* (M. Bieliková, G. Friedrich, G. Gottlob, S. Katzenbeisser, and G. Turán, eds.), (Berlin, Heidelberg), pp. 115–129, Springer Berlin Heidelberg, 2012.

[51] S. Tratz, A. Sanfilippo, M. Gregory, A. Chappell, C. Posse, and P. Whitney, “Pnnl: A supervised maximum entropy approach to word sense disambiguation,” in *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval ’07, (Stroudsburg, PA, USA), pp. 264–267, Association for Computational Linguistics, 2007.

[52] Y. S. Chan, H. T. Ng, and D. Chiang, “Word sense disambiguation improves statistical machine translation,” in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, (Prague, Czech Republic), pp. 33–40, Association for Computational Linguistics, June 2007.

[53] D. Yarowsky, “Hierarchical decision lists for word sense disambiguation,” *Computers and the Humanities*, vol. 34, no. 1/2, pp. 179–186, 2000.

[54] H. Schütze, “Automatic word sense discrimination,” *Comput. Linguist.*, vol. 24, pp. 97–123, Mar. 1998.

[55] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll, “Unsupervised acquisition of predominant word senses,” *Computational Linguistics*, vol. 33, no. 4, pp. 553–590, 2007.

[56] S. P. Ponzetto and R. Navigli, “Knowledge-rich word sense disambiguation rivaling supervised systems,” in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, (Uppsala, Sweden), pp. 1522–1531, Association for Computational Linguistics, July 2010.

[57] R. Sinha and R. Mihalcea, “Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity,” in *International Conference on Semantic Computing (ICSC 2007)*, pp. 363–369, Sep. 2007.

[58] E. Agirre, O. L. De Lacalle, and A. Soroa, “Knowledge-based wsd on specific domains: Performing better than generic supervised wsd,” in *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI’09, (San Francisco, CA, USA), pp. 1501–1506, Morgan Kaufmann Publishers Inc., 2009.

[59] R. Mihalcea, *Knowledge-Based Methods for WSD*, pp. 107–131. Dordrecht: Springer Netherlands, 2006.

[60] R. Navigli and M. Lapata, “An experimental study of graph connectivity for unsupervised word sense disambiguation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 678–692, 2010.

[61] D. Yarowsky, “Unsupervised word sense disambiguation rivaling supervised methods,” in *33rd Annual Meeting of the Association for Computational Linguistics*, (Cambridge, Massachusetts, USA), pp. 189–196, Association for Computational Linguistics, June 1995.

[62] W. A. Gale, K. W. Church, and D. Yarowsky, “One sense per discourse,” in *Proceedings of the Workshop on Speech and Natural Language*, HLT ’91, (Stroudsburg, PA, USA), pp. 233–237, Association for Computational Linguistics, 1992.

[63] D. Yarowsky, “One sense per collocation,” in *Proceedings of the Workshop on Human Language Technology*, HLT ’93, (Stroudsburg, PA, USA), pp. 266–271, Association for Computational Linguistics, 1993.

[64] R. Mihalcea, “Co-training and self-training for word sense disambiguation,” in *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, (Boston, Massachusetts, USA), pp. 33–40, Association for Computational Linguistics, May 6 - May 7 2004.

[65] T. P. Pham, H. T. Ng, and W. S. Lee, “Word sense disambiguation with semi-supervised learning,” in *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI’05, pp. 1093–1098, AAAI Press, 2005.

[66] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell, “Text classification from labeled and unlabeled documents using em,” *Machine Learning*, vol. 39, pp. 103–134, May 2000.

[67] H. Shinnou and M. Sasaki, “Unsupervised learning of word sense disambiguation rules by estimating an optimum iteration number in the EM algorithm,” in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 41–48, 2003.

[68] J. Chen and M. S. Palmer, “Improving english verb sense disambiguation performance with linguistically motivated features and clear sense distinction boundaries,” *Language Resources and Evaluation*, vol. 43, pp. 181–208, Jun 2009.

[69] M. Lapata and C. Brew, “Verb class disambiguation using informative priors,” *Comput. Linguist.*, vol. 30, pp. 45–73, Mar. 2004.

[70] W. Wagner, H. Schmid, and S. S. Im Walde, “Verb sense disambiguation using a predicate-argument-clustering model,” in *Proceedings of the CogSci Workshop on Distributional Semantics beyond Concrete Concepts*, pp. 23–28, Citeseer, 2009.

[71] D. Dligach and M. Palmer, “Novel semantic features for verb sense disambiguation,” in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short ’08, (Stroudsburg, PA, USA), pp. 29–32, Association for Computational Linguistics, 2008.

[72] G. Christie, A. Laddha, A. Agrawal, S. Antol, Y. Goyal, K. Kochersberger, and D. Batra, “Resolving language and vision ambiguities together: Joint segmentation & prepositional attachment resolution in captioned scenes,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 1493–1503, Association for Computational Linguistics, Nov. 2016.

[73] X. Chen, A. Ritter, A. Gupta, and T. Mitchell, “Sense discovery via co-clustering on images and text,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5298–5306, June 2015.

[74] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, pp. 886–893 vol. 1, June 2005.

[75] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, Sep. 1999.

[76] A. Di Marco and R. Navigli, “Clustering and diversifying web search results with graph-based word sense induction,” *Computational Linguistics*, vol. 39, no. 3, pp. 709–754, 2013.

[77] O. Melamud, J. Goldberger, and I. Dagan, “context2vec: Learning generic context embedding with bidirectional lstm,” in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 51–61, 2016.

[78] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, vol. 3rd edition draft. -, 09 2018.

[79] X. Rong, “word2vec parameter learning explained,” *ArXiv*, vol. abs/1411.2738, 2014.

[80] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, vol. 14, pp. 1532–1543, 01 2014.

[81] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, “Advances in pre-training distributed word representations,” in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[82] H. Hotelling, “Relations between two sets of variates,” *Biometrika*, vol. 28, pp. 321–377, 12 1936.

[83] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, “Canonical correlation analysis: An overview with application to learning methods,” *Neural Computation*, vol. 16, no. 12, pp. 2639–2664, 2004.

[84] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, “Deep canonical correlation analysis,” in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 1247–1255, PMLR, 17–19 Jun 2013.

[85] F. Yan and K. Mikolajczyk, “Deep correlation for matching images and text,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3441–3450, June 2015.

[86] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proc. of NAACL*, 2018.

[87] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[88] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.