Loopy Belief Propagation for Large-Scale Collective Entity Linking

Master Thesis
Marina Horlescu
July 15, 2014

Advisor: Prof. Dr. Thomas Hofmann
Department of Computer Science, ETH Zürich
Abstract

Entity linking, a very popular research topic nowadays, involves identifying mentions of ‘real world’ entities in natural language text and associating them with a representation from a knowledge thesaurus. It is a form of semantic grounding of discourse in a background repository of world knowledge and therefore a key step towards deep language understanding. Applications of this research area include semantic search, automatic document enrichment, ontology learning, relation extraction, question answering, etc.

An entity linking pipeline deals with two problems: first, identifying token spans that should be linked and, second, selecting the corresponding entity from an entity catalogue. Traditionally, local approaches that make independent entity linking decisions have been proposed. They are largely based on the local compatibility between the context of the mention and the textual metadata of the entity. These techniques are limited because they do not make use of the underlying entities that co-occur in the text. Alternatively, collective entity linking is a better solution that resolves all names in a document simultaneously. This is motivated by the fact that in a coherent document co-occurring entities are typically semantically related. Inferring the best global assignment of entities requires searching through an exponential number of entity resolution combinations. The resulting MAP inference problem can be NP-hard and therefore approximations or pruning techniques are used.

In this thesis we propose solving this MAP inference problem via loopy belief propagation for entity disambiguation. We model the joint distribution through log-linear models and compare with state of the art techniques. We assume that frequent co-occurrence of entity pairs implies semantic relatedness. We then use collected statistics from a corpus to make an informed decision. Exploiting calibrated pairwise and single node potentials leads to better results while comparing to a recently proposed algorithm: a random graph walk method.
Acknowledgements

I want to sincerely express my gratitude to my supervisor, prof. Thomas Hofmann, for giving me the opportunity to work on a topic that I found very interesting. He is an excellent professor and I thank him for his insightful guidance, substantial contributions to this work, valuable feedback and for his constant positive attitude.

I have really enjoyed working in the new Data Analytics Lab and for this I am grateful to everyone in the group, who made me feel part of the team. I also thank them for their useful feedback given during our weekly meetings.

I wholeheartedly thank Octavian Ganea for his strong involvement and key ideas that lead to improvements in my work.

I warmly thank Carsten Eickhoff for his constant involvement and support, for sharing his ideas and for his guidance during my first months, and, together with Octavian Ganea and Aurelien Lucchi for carefully proofreading my thesis and offering substantial feedback.

I reserve special thanks to Octavian for colouring my life during my time at ETH and for being a constant source of encouragement.

Most importantly, I thank my family for always motivating me and being by my side, despite the physical distance.
# Contents

## 1 Introduction
1.1 Problem statement ........................................... 1  
1.2 Motivation and Challenges ................................. 3  
1.3 Thesis Goals and Contributions ............................. 5  
1.4 Outline .......................................................... 5  

## 2 Related Work
2.1 Local methods .................................................. 7  
2.2 Global methods ................................................ 8  
2.3 Graph based methods .......................................... 9  
2.4 Topic modelling methods ....................................... 10  

## 3 Entity Linking
3.1 Wikipedia processing .......................................... 11  
3.1.1 Indices construction ......................................... 13  
3.1.2 Scalable implementation of the indices construction .... 14  
3.2 Mention Detection .............................................. 15  
3.3 Entity Disambiguation ......................................... 17  
3.4 Resources ...................................................... 18  
3.4.1 Knowledge Bases ............................................ 18  
3.4.2 Corpora ..................................................... 19  
3.4.3 Testing collections ......................................... 19  

## 4 Baseline methods
4.1 Random Graph Walk ........................................... 21  
4.1.1 Referent Graph ............................................ 22  
4.1.2 Collective entity linking ................................... 24  
4.2 Most Frequent Entity ........................................... 26
## Contents

5  Loopy Belief Propagation 27
   5.1 The underlying graphical model 27
   5.2 Inference in Pairwise Markov Random Fields 31
      5.2.1 The Max-Sum Algorithm 32
      5.2.2 Loopy belief propagation 36
   5.3 Joint probability models for entity linking 38
      5.3.1 Uncalibrated model 39
      5.3.2 Calibrated model 40

6  Experiments and results 49
   6.1 IITB Dataset 49
   6.2 Evaluation metrics 50
   6.3 Experimental results 50
      6.3.1 Mention Detection Performance 50
      6.3.2 Entity Disambiguation Performance 53
      6.3.3 Overall Entity Linking Performance 55

7  Conclusion and Future work 57

Bibliography 59
Chapter 1

Introduction

Entity linking is the task of identifying mentions of entities in natural language text and of linking these mentions to the corresponding entry in a knowledge base of entities.

Examples of entities include, but are not limited to: people, places, organizations, books, songs, events and competitions. Also, in general, any noun phrase can correspond to an entity (e.g. ‘cold ironing’\(^1\)).

Entity linking can be applied on various forms of text: from long and well structured documents, like news and blog posts, to short and poorly composed texts, such as tweets and search engine queries.

Natural language semantics requires understanding the meaning of individual language elements. Entity linking achieves the goal of attaching semantics to mentions of ‘real world’ entities and is a key step towards natural language understanding.

Historically, entity linking is preceded by named entity recognition, a related task, that seeks to detect and classify certain text elements into predefined categories: people names, organization names, time expressions, monetary values, etc. The main difference between the two tasks is that entity linking, in addition, tries to associate an entity entry from a knowledge base to the identified text element.

In the next section, we give an entity linking example and define terminology used throughout this thesis.

1.1 Problem statement

Given a document, we want to annotate occurrences of entities in the text. Let us consider that the entity universe is represented by the set of Wikipedia

\(^1\)See http://en.wikipedia.org/wiki/Cold_ironing for concept definition.
1. Introduction

articles. In the following snippet we give a solution for a short piece of text.

Müller [Thomas Müller], a player [Football player] of Germany [Germany national football team], helped his team to reach the final [2014 FIFA World Cup Final] of the cup [2014 FIFA World Cup] by scoring one goal against Brazil.

The words in italics are identified names that refer to entities. Following is the entity annotation in square brackets. Each detected name is annotated with the title of its corresponding Wikipedia page.

The annotations in the example are not exhaustive to keep the example text uncluttered.

The task is not trivial because of the name ambiguity problem. Each entity may have multiple surface forms and each surface form may refer to multiple entities. For example, the entity ‘Thomas Müller’ can have as surface forms both ‘Thomas Müller’ and ‘Müller’. Conversely, ‘Müller’ can be the surface form of multiple entities: the German footballer, the former German skier, the Swiss politician and others.

Furthermore, mentions that refer to entities not covered by the entity repository should not be linked.

Let us define the terminology and notations used throughout this thesis.

A knowledge base is the background repository of entities. It can contain descriptions of the entities and relations between them. In our case, it is the English version of Wikipedia. Available alternatives include DBpedia, Freebase and YAGO, that are described in Section 3.4.1.

An entity is uniquely represented by the title of an English Wikipedia article. Examples of entities in the above text are $e_1$ = ‘Thomas Müller’ and $e_2$ = ‘Football player’.

We will use the notation $\mathcal{E}$ for the entity universe, that is the set of all possible entities.

A mention is a token sequence that is potentially a reference to an entity. In the above text, we identify mentions like ‘Germany’ and ‘final’. We can also refer to a mention as a surface form. We denote by $M$ the set of all mentions in a document.

The context of a mention is the surrounding text in a suitable window around the mention.

**Formal problem statement:** Given a document $D$, an entity linking pipeline finds a set of annotations $\{(m_i, e_i)\mid i \in 1, n\}$ where the entity $e_i \in \mathcal{E}$ is the correct annotation of mention $m_i$. Each mention $m$ is represented by its token span $m.name$ and offset where it occurs in the text $m.offset$. 

Our approach is to perform **collective entity linking**, by making each entity linking decision for a mention, based on all other entity linking decisions taken for the document.

Next, we present a series of applications that motivate us in pursuing the entity linking task. Moreover, we emphasize the key challenges that have to be overcome in order to solve the problem.

## 1.2 Motivation and Challenges

Entity linking has a number of direct applications. Alternatively, it can be used as a feature in pipelines performing more complex natural language processing tasks.

**Automated document enrichment** is an immediate application of entity linking. For example, a highly technical text can be made easier to understand by providing references to new concepts, in the same manner as hyperlinks provide additional information for Web pages. The reader can then follow hyperlinks to read more information on concepts he wants to understand better. In educational applications, this would grant students easy access to additional information relevant to the studied topic.

**Semantic search** builds upon the benefits of understanding searcher intent and the contextual meaning of terms and seeks to improve accuracy with relevant results. Entity linking can give strong cues on certain terms’ semantics and can gear search results by understanding better the user’s intent.

**Ontology learning** can be enabled via entity linking. Extracted entities from text can be used together with extracted relations and structure to populate an ontology.

In addition, entity linking can also be used as a feature in pipelines dealing with **text classification, word sense disambiguation, measuring semantic similarity, sentiment analysis, automatic document summarization, question answering**, etc.

In the process of entity linking a series of **key challenges** have to be solved:

- **Pruning the search space**: Investigating all entities in the entity universe as possible candidates for a mention is inefficient. We want to consider only relevant candidates for a given mention name. We use anchor texts as cues of ways to refer to an entity. Even by exploiting anchor texts, the number of candidate entities can still be fairly large. For example, the name ‘Michael’ is linked to 395 pages throughout Wikipedia, out of which only 7 pages are appearing in more than 2% of all links with anchor text ‘Michael’. We filter out infrequent candidate entities that have an occurrence probability below a threshold,
1. Introduction

inspired by the work in [18]. This improves precision and slightly reduces recall, with an overall better F1-score. The running time is greatly reduced because the search space is pruned.

Collectively disambiguating all mentions in the text widens the search space even more, making an exponential number of entity resolution combinations possible. Since the resulting MAP-inference problem is NP-hard, we use approximate inference. Loopy belief propagation, an approximate inference method, has been shown to have good results in certain domains [20]. Employing loopy belief propagation results in a solution with polynomial time complexity. For these reasons we have decided to use this algorithm in this work.

• **Deciding which mentions to link**: We want to filter out terms that do not correspond to entities: for instance, most verbs and adjectives. Moreover, low interest entities may not be linked. For example, even punctuation marks have Wikipedia pages, like colon or semicolon. Also, we want to abstain from linking entities that are not covered by Wikipedia.

For this purpose, we use in this thesis, a filtering by keyphraseness scores [16]. This score measures how often a name is linked to any entity as opposed to how often it is not linked at all. We compute the scores over a Web corpus, for example in Wikipedia. We vary a threshold to handle how many entities we link, effectively controlling the precision-recall trade-off.

In our approach, we exclude mentions that we decided not to link, at the beginning of the algorithm, and link all remaining mentions to the best fitting candidates.

Accurately deciding what to link or not link seems to be a much harder problem. For example, it is hard to refrain from linking a person name to Wikipedia, when the person does not have a Wikipedia page, but a different person with the same name has. This region of the algorithm can further be improved in the future.

• **Disambiguating a mention**: Once we decided to link a surface form, we need to distinguish the correct entity from a list of potential candidates. Our approach is to use valuable information regarding the disambiguation of other entities appearing in the same document. We collect statistics on co-occurrence of entity pairs and integrate these in a model of the joint probability. We then use loopy inference to compute the best entities assignment. We make incremental improvements to the joint probability model that eventually outperforms both baselines used for comparison.
1.3 Thesis Goals and Contributions

We compare with a ground truth dataset designed to annotate all possible token spans that have a corresponding Wikipedia article. As a consequence, we aim to annotate exhaustively.

Here is a summary of the key contributions of this thesis:

- **Novel solution**: We aim to collectively resolve all mentions appearing in a document and choose the best global entity assignment. Using loopy belief propagation is a novel approach, well suited for the task as it can model interdependencies of entities while still resulting in a polynomial solution.

- **Calibrated joint distribution**: We use log-linear models with pairwise and single node potentials to explain the joint distribution of the entities. We compute pairwise weights calibrated on an observed corpus. We compare this model with an ad-hoc joint probability model and show how calibration improves performance.

- **Scalability and performance**: We use a Map-Reduce framework that learns statistics from the Wikipedia dataset in a distributed manner. Annotating documents of reasonable size is also fast, and can be scaled by splitting the document collection over shards. For example, our dataset of 100 documents, with a set of 10,000 identified mentions required approximately 5 minutes to be linked with serial processing on a single processing unit.

  We empirically show that our method outperforms a current alternative based on a random graph walk [7]. At the same time, our method is better than a simple greedy baseline, one that is even more efficient than the random graph walk technique.

1.4 Outline

We provide here an outline of the main points of the following work.

In the second chapter, we present related work on entity linking to give a global view on the work in the field. Entity linking methods can be grouped in four categories: local context, global context, graph based and topic modelling solutions. We summarize a few papers from each category.

The third chapter outlines the steps of the entity linking process: mining the Wikipedia dataset, performing mention detection and entity disambiguation. Specifically, we illustrate the construction of precomputed indices from Wikipedia, we report the method that we use for mention detection and the general approach for entity disambiguation. We also make a short overview
1. **Introduction**

of available resources: knowledge bases, Web corpora and testing collections.

The fourth chapter explains in detail two used baselines: a random graph walk procedure and a greedy routine, referred as the ‘Most frequent entity’ baseline. Interestingly, the brute-force baseline outperforms the random graph walk technique and has reasonably good results.

Following, the fifth chapter describes the graphical model that we use for loopy belief propagation. We present the max-sum algorithm and its generalization, loopy belief propagation, that we use for inference. Then we propose two models of the joint probability of mentions and entities. The first is inspired by the measures used in the random graph walk. The second calibrates single node and pairwise potentials on a corpus.

The results of our experiments are detailed in the sixth chapter. Here we first discuss aspects regarding the chosen ground truth dataset and the evaluation methods that we adopt. We reveal the performance at each step of the entity linking process and we also show the key improvements that boost the results.

In the last chapter, we draw the main conclusions of our work and suggest future improvements.
In this chapter, we will briefly present state-of-the-art entity linking techniques. An excellent summary of entity linking work is done in [14].

As from [14], current approaches are grouped in: local methods that aim to disambiguate each mention independently, global methods that collectively solve the task, graph based methods and topic modelling approaches.

Our method can be classified into the global approach category, as we aim to choose the best global assignment over all possible entity resolution combinations. We present a few papers from each category to give a global view on the work in the field.

2.1 Local methods

Local based approaches make independent linking decisions for each identified mention. Typically, discriminative features are extracted from the local context of a mention, for example important words. Similarly, for each entity, discriminative features can be extracted from its Wikipedia page. A mention is linked by comparing its context features to the feature representation of each candidate entity and choosing the best match.

The main drawback of this approach is that valuable information about all other entities appearing in the same document is unused. We expect that the entities mentioned together in a topically coherent document are semantically related.

The paper [16] compares three scoring methods for mention detection: tf-idf, $\chi^2$ independence test and keyphraseness. The mentions scoring below a threshold are not linked. Keyphraseness, that we also use for mention detection, significantly outperforms the other two. However, there is still room for improvement as it achieves only 54.63% $F_1$ score. Two approaches are
2. Related Work

tried for entity disambiguation. The first method picks the candidate with largest overlap between the paragraph of the mention and the candidate’s page. This performs worse than a baseline that chooses the most likely candidate. The second method is a naive Bayes approach using as features: the mention and neighbouring words, the part of speech of the mention and of the surrounding words and entity specific terms. Finally a voting scheme that disambiguates only when both methods agree is employed. This trades off somewhat lower recall for a higher disambiguation precision.

Other work in this category includes [18] that starts with detecting unambiguous mentions and matching them to the unique candidate. All other mentions are linked depending on their relatedness to the unambiguous mentions. Each unambiguous mention is assigned a weight: its keyphraseness score. A semantic relatedness measure is used to compare entity pairs. Then the score of a candidate entity is the weighted average of relatedness between the candidate and each unambiguous mention. This score, \( P(\text{name|entity}) \) and the context quality – the sum of weights assigned to unambiguous mentions, are used as features. The best performing classifier is a bagged C4.5. Afterwards, mentions that should not be linked are filtered out by a classifier. The features used are: keyphraseness score, relatedness, the disambiguation confidence, generality (candidate article depth in Wikipedia’s category tree), spread (the frequency of the mention in the document) and location. The location feature highlights mentions appearing in the introduction or conclusion; they are more likely to be linked. The disadvantage of this approach is that it relies on a small fraction of unambiguous mentions that hopefully would have high disambiguation utility.

2.2 Global methods

Global methods disambiguate all mentions in a document collectively, by choosing the assignment that best fits the entire document. The search space grows exponentially with the number of mentions and optimizations are needed to reduce the search space. Our method is included in this category.

In the paper [3], surface forms are detected via a named-entity-recognizer based on rules and corpus statistics. Each entity is assigned a vector space representation of two components: Wikipedia categories of its article and a pruned set of links from the page: all links from the first paragraph and reciprocal links, that are links towards articles that link back to the page. Ideally, this paper wants to choose entities that maximise agreement between their vector space representations: that largely have close values of their vector space representations. For each mention, an entity is chosen that maximizes agreement with all other candidate entities of all other surface forms. Inevitably, irrelevant candidates influence the outcome of the entity
2.3. Graph based methods

linking process.

The work in [11] models the joint probability of the entities based on the local compatibility between each mention and its corresponding entity and on the topical coherence between all entities. Finding the best entity assignment for a fixed set of surface forms is NP-hard and two approximation methods are tried in this paper: linear programming and hill-climbing. The first outperforms the second, but has the disadvantage that it is slower.

In [23], a stage of machine learning for entity disambiguation is followed by a stage of machine learning for filtering out mentions that should not be linked. In the first stage, both local and global features are used: cosine similarity between mention context and entity page representations, Pontwise Mutual Information and Normalized Google Distance – average or maximum over entity pairs. In the second stage, several features are added on top: entropy distribution of $P(\text{name}|\text{entity})$ and others. SVMs are used in both stages as the preferred classifiers.

The system presented in [4] focuses on very short documents, composed of a few tens of words: tweets, snippets of text, etc. A voting scheme is used: each mention votes for each candidate of a distinct mention. The candidate with highest score cumulated from votes wins. The voting score makes use of information regarding semantic relatedness between entities, the probability of an entity given the name and the total number of candidates for an anchor. Mentions are not linked if they have low keyphraseness and the assigned entity is not coherent with the other identified entities.

2.3 Graph based methods

Graph based methods model the entity linking problem in a graph setting and operate with graph methods: PageRank, random graph walk, etc.

One of the baselines that we use is included in this category: the random graph walk method. See Section 4.1 for the description of the method.

The work of [5] explores two graph methods: degree centrality (ratio of edges adjacent to a node) and PageRank. The graph nodes are articles and article categories. The size of the graph is limited by only exploring vertices at limited distance to the candidate nodes. Each candidate is scored via page rank or degree centrality. This score is then weighted by the cosine similarity between the local context of the mention and the entity’s page. The candidate with maximum score is chosen for each entity.

Another graph-based paper that we do not detail here is [21]. Its goal is entity linking on closed captions. The solution works with real-time streaming data and obtains high-precision.
2.4 Topic modelling methods

Recent work has applied topic modelling approaches for solving the entity linking problem.

Latent Dirichlet allocation is used in [22] and [6].

A semi-supervised topic modelling solution is given in [9].
Chapter 3

Entity Linking

In this chapter, we examine the general steps of an entity linking pipeline. We describe in detail our solution for the first two stages of the pipeline. The final entity disambiguation phase is discussed in detail for the baselines and our method in the next two chapters.

In the last section, we make an overview of alternative resources that could be used for future work: knowledge bases, Web corpora and test collections.

Entity linking is typically performed in three stages:

- **Preprocessing**: frequency statistics are collected from a corpus, for example: co-occurrence frequency of entity pairs; frequency of hyperlinks with a given anchor text and a given entity expressed by the URL of the hyperlink. These statistics will be used in the following two phases.

- **Mention detection**: mentions are identified and mentions that should not be linked are filtered out.

- **Entity disambiguation**: each mention is disambiguated to an entity chosen from a list of candidates.

Filtering out mentions that should not be linked can be done before and/or after entity disambiguation. For instance the final scores obtained in the disambiguation stage can be used for a final filtering of the linked mentions. In our approach, we decide which mentions will be linked before the entity disambiguation phase.

### 3.1 Wikipedia processing

Wikipedia is a free-access encyclopedia written by volunteers worldwide. It can be seen as a repository of entities where each article corresponds to
3. Entity Linking

An entity. The size of English Wikipedia offers a vast coverage of around 4,500,000 articles. This is the knowledge base of choice for our work.

An entity can be uniquely represented by its article title. The title is also expressed in the URL. English Wikipedia URLs are of the form ‘http://en.wikipedia.org/wiki/(title)’, where (title) is the title of the article. Note that spaces can be interchanged with underscores in the URL.

There are special types of Wikipedia pages used to organize Wikipedia:

- **Redirect pages** do not have any content, they only redirect the users to a canonical page. We handle the title of the redirect page as a possible surface form for the entity represented by the canonical page.

- **Disambiguation pages** have the purpose of organizing articles that refer to distinct entities with a common name. The entities appearing in the disambiguation page are usually not semantically related. For example the disambiguation page ‘Columbia’ lists the Space Shuttle Columbia, Columbia Pictures and Columbia University.

- **List pages** organize articles that have a common category or trait, for example the article ‘List of bookstore chains’.

We consider that these special types of articles do not represent canonical entities. After filtering out these articles we get a set of 3,796,235 canonical entities.

To get us started, there are two types of information that we need to mine from intra-Wikipedia hyperlinks:

- Possible surface forms of entities: anchor texts can be leveraged as surface forms for the entities corresponding to the anchor targets.

- Statistics on entities and entities pairs: we gather information on frequencies of entities and entities pairs that we will use at a later stage.

For these purposes we can construct two indices:

- **Anchor text index**: maps each surface form to a list of candidate entities.

- **Entity links index**: maps an entity to a list of other articles that link to it.

Next, we explicitly describe the construction steps of the anchor text index and the entity links index. In a similar manner, we gather other types of information needed for later stages of the entity linking pipeline. We store this data in indices for fast retrieval and efficient query time.

---

1Throughout this thesis, we use the English Wikipedia version of February 2014
3.1. Wikipedia processing

3.1.1 Indices construction

The anchor text index is stored in a hash map and contains a set of names, that are possible ways of referring to an entity in text. Each name in the index is assigned a list of candidate entities. We also store the occurrence frequency for each name–entity pair.

The entries in the anchor text index are mined from Wikipedia hyperlinks. Given a hyperlink that points to a Wikipedia article, its creator has chosen a meaningful anchor text that refers to the entity represented by the anchor URL. We consider that the anchor text is a possible surface form for the target entity. For each anchor text, we gather a list of candidate entities from the hyperlinks occurring in Wikipedia.

Typically, the distribution of candidate entities for a given name has a long tail. In other words, there are many entities that are linked infrequently under a given name. Inspired by the work in [18], we keep only candidate entities that appear in more than 2% of all links with a given anchor text. This improves the running time because the search space is reduced. At the same time, precision and recall also improve, because many irrelevant candidates are not considered. However, the maximum achievable recall is lowered, because candidates from the anchor text index, that may be correct, are discarded. The impact of this change is presented in Table 6.1.

The steps of the construction of the anchor text index:

1. Iterate over anchor pairs (anchor text, anchor url). Follow steps 2-5 for each anchor.

2. Discard anchors that point to non-canonical articles: redirect pages, disambiguation pages, list pages or non-existing articles – the red links.

3. Normalize anchor url: unescape XML and HTML characters. If needed, the URL is decoded and underscores are replaced with spaces.

4. Normalize anchor text: unescape XML and HTML characters, lower case, remove remaining HTML tags, remove consecutive whitespaces, remove uninformative punctuation, but keep delimiters like ‘:.’.

5. The title of the target article is extracted from the anchor url. Then, it is added in the index to the candidates list of the anchor text. If the title already exists in the index, its frequency is incremented by one.

6. Each title of a redirect page is added to the index as a surface form of the entity that corresponds to the canonical page pointed by the redirect.
3. Entity Linking

7. Candidate entities of a given name are discarded, if they appear in less than 2% of the hyperlinks with the anchor text equal to the given name.

The entity links index stores in a hash map, the list of entities that contain a link in their article to a given entity. In other words, an entity is mapped to the list of entities that link to it, in their Wikipedia articles.

The steps of the entity links index construction:

1. Iterate over Wikipedia articles. Follow next steps for each article.
2. If the article is not canonical we can discard it because the mentioned entities are usually not semantically related. Otherwise, we continue with the next steps.
3. Iterate over all hyperlinks in the article and follow next steps for each hyperlink.
4. Normalize anchor url: unescape XML and HTML characters. If needed, the URL is decoded and underscores are replaced with spaces.
5. The title of the article that contains the link is added in the index, to the list of articles that link to the anchor target.

To reduce the memory footprint, we assign each Wikipedia entity a unique integer ID and in all indices we will then store entities by their integer IDs rather than titles.

Measuring entity co-occurrence is a key step in our pipeline. We need to efficiently answer how many documents link to two specified entities. For this reason, we store in the entity links index, the posting list of each entity, sorted increasingly by the IDs of the entities. At query time, we retrieve each sorted list of articles that mentions each entity of the pair. Then, we compute the intersection on the fly by a single iteration over the posting lists, in linear time in the number of in-links.

Our solution is also adequate when we apply it on a large corpus of Web pages to collect entity statistics. This is enabled by an optimized distributed implementation. The next section discusses the important tools that efficiently process Wikipedia and the Map-Reduce solution that distributes the work.

3.1.2 Scalable implementation of the indices construction

The latest English Wikipedia dump can be downloaded in archived XML format. We use the Wikipedia version of February 2014.

\[\text{http://download.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz}\]
3.2. Mention Detection

Extracting the plain text article content needs XML parsing and specific Wikipedia parsing. For example, intra-Wikipedia hyperlinks are stored in the dump in a Wikipedia specific format: [[target article title | displayed text]]. When the displayed name is the same as the target article title, the format [[target article title]] can be used. A convenient, fast Wikipedia parser is Bliki \(^3\). The parser enables plain text retrieval of a Wikipedia article from the XML dump. From the plain text, we can extract statistics that we will use later in the pipeline, like document frequencies of terms occurring in Wikipedia.

Building the anchor text index and the entity links index can be done efficiently in a distributed Map-Reduce setting. Cloud\(^9\) is an excellent Hadoop toolkit \(^4\) that can be used for working with Wikipedia in a distributed Map-Reduce fashion. It is integrated with the Bliki parser.

With the Cloud\(^9\) toolkit, we repack the XML dump into compressed sequence files of reasonably sized blocks. Each block will be fed to one mapper; having multiple blocks will allow for the data to be spread out. Repacking the input data will utilize less space, will allow faster computations and will enable distributed processing by spreading the data to multiple mappers. The toolkit also provides utilities for extracting the anchors from an article and retrieving the article’s plain text content.

We also extract a mapping of redirect pages to the canonical pages they point to. For this, we use a toolkit called wikipedia-redirect \(^5\).

By using a Map-Reduce solution and by repacking the input in a convenient number of sequence file blocks, we can scale the solution depending on the number of available processing units. This means that we can always reduce the running time by adding more machines, the lower bound being the maximum time for processing one Wikipedia page, which is in the order of seconds.

The indices built by processing Wikipedia are stored and will be used in the next two entity linking steps: mention detection and entity disambiguation.

### 3.2 Mention Detection

Given a document collection on which we want to perform entity linking, we first perform mention detection and later entity disambiguation on each of the documents. This structure of the solution is used for our method and also for our two baselines. We use the same mention detection solution for all three pipelines, yet, we use distinct entity disambiguation solutions.

\(^3\)http://code.google.com/p/gwtwiki/

\(^4\)http://lintool.github.io/Cloud9/index.html

\(^5\)https://code.google.com/p/wikipedia-redirect/
3. Entity Linking

The **goals** of the mention detection are:

- To identify potential mentions of entities in the input text.
- To associate a set of candidate entities to each mention.
- To filter out the mentions that should not be linked.

The key step is filtering out mentions that should not be linked. A commonly used method is filtering by keyphraseness score. In the work [16], three mention detection methods are compared: keyphraseness, tf-idf and \( \chi^2 \) independence test. Keyphraseness outperforms the other two significantly, but there is still room for improvement as the obtained \( F_1 \) score is 54.63% on a test collection of 85 Wikipedia pages.

**Keyphraseness** measures how often a name is linked in a web corpus. In other words it tries to estimate the probability that a hyperlink to a Wikipedia page will be created for a given name. Formally:

\[
\text{keyphraseness}(n) = \frac{|D_{n,\text{link}}|}{|D_n|}
\]

where \( n \) is a mention, \( D_{n,\text{link}} \) is the set of documents that contain at least one link with \( n \) as anchor text and \( D_n \) is the set of documents that contain the mention \( n \), with or without a hyperlink.

Keyphraseness scores may become unreliable for rare mentions. We address this problem by discarding all mentions that appear less than five times in Wikipedia. This lowers the number of names appearing in the anchor text index, from around 9 million names, to about 5,1 million names.

The **mention detection procedure** is applied as follows on an input text:

1. The text is tokenized and n-grams are gathered. We gather n-grams of up to 11 tokens. The maximum number of tokens was chosen such that no annotation in our ground truth dataset has more tokens.

2. The n-grams are searched in the anchor text index of Wikipedia and their list of candidate entities is retrieved from the index. The n-grams that do not exist in the index are filtered out. A list of mentions and their candidate entities is maintained.

3. Mention that have keyphraseness score lower than a threshold are filtered out. This threshold can be varied, effectively trading recall for precision.

4. Output: mentions with a list of candidate entities for each mention.

Efficient computation is enabled by a preprocessing of the keyphraseness scores and storing them in the anchor text index together with the candidate entities of each mention.
Using keyphraseness obviously has drawbacks: if a name is linked at least once, there will be contexts where it should be linked.

### 3.3 Entity Disambiguation

Entity disambiguation is the main focus of this work. It can be stated formally in the following terms. The input is a document that is associated with a set of detected mentions $M = \{m_1, m_2, \ldots, m_N\}$, where each mention $m$ has a list of candidate entities $m.E \subseteq E$, a name $m.name$ and a document offset where it occurs $m.offset$. The goal is to produce a tuple of entities $(e_1, e_2, \ldots, e_N)$ such that entity $e_i$ is the correct disambiguation of mention $m_i$.

Local approaches try to find entities that maximize a local criteria for each mention–entity pair. Let $\phi(m_i, e_i, c_i)$ be a scoring function that measures how likely entity $e_i$ is the correct disambiguation of mention $m_i$ that occurs in the context $c_i$. The local context $c_i$ of the mention can be, for example, the set of words surrounding the mention. The optimization problem solved by local approaches is:

$$e_i^* = \arg \max_{e_i \in m_i.E} \phi(m_i, e_i, c_i), \ \forall i \in \{1, N\}$$

Our most frequent entity baseline is a local approach with:

$$\phi(m_i, e_i, c_i) = p(e_i|m_i)$$

Global approaches attempt to maximize both local criteria and an overall agreement between the assigned entities. Let’s denote a configuration of the entities occurring in a document by: $\Gamma = \{(e_1, e_2, \ldots, e_N)| e_1 \in m_1.E, e_2 \in m_2.E, \ldots, e_N \in m_N.E\}$. Let $\psi(\Gamma)$ be a scoring function that measures how likely the $\Gamma$ assignment is the correct disambiguation of the set of mentions $M$. The resulting optimization problem finds the best entities assignment $\Gamma^*$, such that:

$$\Gamma^* = \arg \max_{\Gamma} \psi(\Gamma) + \sum_{i=1}^N \phi(m_i, e_i)$$

Typically, modelling the collective entities topical coherence is easily done by assuming $\psi(\Gamma)$ can be computed from pairwise scoring functions $\psi(e_i, e_j)$. Then, the easier optimization problem is solved:

$$\Gamma^* = \arg \max_{\Gamma} \sum_{1 \leq i < j \leq N} \psi(e_i, e_j) + \sum_{i=1}^N \phi(m_i, e_i)$$

Our method solves this optimization problem by applying loopy belief propagation and by calibrating the scoring functions $\psi(e_i, e_j)$ on observed data.
We have detailed our use of Wikipedia as a knowledge base and Web corpus. In the next section we briefly go through alternative resources that can be used for entity linking. Exploiting a larger Web corpus can be a future work direction.

3.4 Resources

The ingredients for a full entity linking system are: a knowledge base, a corpus for extracting frequency statistics and a test collection. In this section, we give an overview of available alternatives.

3.4.1 Knowledge Bases

A knowledge base provides a representation for the entity universe that we want to link to. Several knowledge bases are available, for example: Wikipedia, Freebase, DBpedia and YAGO. They differ in coverage, some provide entity description and relations between entities, some offer multiple language coverage or are domain-oriented.

DBpedia [12] is a crowd-sourced community project that provides structured information from Wikipedia in multiple languages. The information extracted from the English Wikipedia comprises around 4 million entities, out of which 3.22 million are classified in an ontology. This ontology groups entities into categories like people, places, creative works, organizations, species, diseases, etc. Together with the other 118 languages, the full DBpedia features 12.6 million entities.

Freebase is a repository much larger than Wikipedia, as it features 39 million entities organized in a graph. The entities are annotated with types and properties. It also provides relations between entities. Initially, Freebase was seeded from high-quality open data. Currently, it is mainly composed by the community. It harvests many data sources including Wikipedia, MusicBrainz and others. The domain of the entities in Freebase is geared towards entertainment. For more than 85% of the entities no text is available, although some of the entities have a link to their respective Wikipedia article.

YAGO is a thesaurus derived from Wikipedia, WordNet and GeoNames. It comprises more than 10 million entities and 120 million facts about them. The accuracy of YAGO is 95% according to manual evaluation. Relations are annotated with confidence scores. Temporal and spacial dimensions are attached to many facts and entities contained. Entities are assigned to more than 350,000 classes that combine Wikipedia categories and the WordNet taxonomy. In addition, thematic domains from WordNet are available, like ‘music’ or ‘science’.
3.4. Resources

3.4.2 Corpora

Wikipedia hyperlinks from Web corpora can be mined to obtain entity statistics. Anchor texts provide ways of referring to an entity, and entity co-occurrence is a cue for entity relatedness. Available corpora include Wikipedia itself,Wikilinks [24], ClueWeb and Crosswikis [25].

**Wikilinks** is a corpus of 10 million Web pages containing 40 million links to 3 million distinct entities. Extracted anchors, as well as the full web page content are provided. Hyperlinks are annotated with Freebase IDs, in case one exists.

**Crosswikis** is a resource mapping anchor texts to Wikipedia entities. The data is mined over Web pages including Wikipedia articles. A frequency count is provided for each anchor text and entity pair. Crosswikis can thus be used to estimate $P(e|n)$ and $P(n|e)$, where $n$ is an entity name and $e$ is an entity.

**ClueWeb12** is a massive dataset that contains 730 million Web pages. Annotations to Freebase entities are provided for a set of 450 million Web pages. Pages with no detected annotations are excluded from the dataset.

Because of editing style guides, Wikipedia provides cleaner data. For instance, we do not need to clean up Wikipedia anchors of general placeholder anchor texts like ‘See here’ or ‘Wikipedia’. Also, Wikipedia style guides provide a guarantee for reasonable keyphraseness scores: because important entities are usually linked, we can rely on linkage frequency for mention detection.

3.4.3 Testing collections

A variety of ground truth datasets are available. They differ in size, coverage and bias. We describe a few of the most used collections in the literature.

**Wikipedia** can be used for testing by trying to predict the intra-Wikipedia hyperlinks. However, style guides of Wikipedia make the encyclopedia biased. Usually an entity should not be linked more than once in a page and the existing links should not be too dense to not clutter the text. Because our goal is to exhaustively link entities in a text, we cannot use this resource for testing.

The **MSNBC** dataset comprises 20 news articles collected for testing by [3] and linked to the English Wikipedia version 2006. The set contains 756 identified surface forms, out of which 127 are marked as non-recallable: no appropriate entity exists in Wikipedia. The data may present bias as it was collected by correcting the output of an implemented system.

---

6Dataset can be obtained from [http://lemurproject.org/clueweb12/FACC1/](http://lemurproject.org/clueweb12/FACC1/)
3. Entity Linking

A subset of the AQUAINT text corpus of 50 newswire stories was used by [18]. Amazon Mechanical Turk was used to annotate the data resulting in 449 links. The annotations mimic the structure of Wikipedia hyperlinks: only the first occurrence of important entities are linked, while uninteresting or redundant mentions of the same entity are not linked.

The work by [23] constructed a test collection from a subset of the ACE co-reference data set. The advantage of this collection is that mentions and their types are provided and co-reference is solved. First mention of each co-reference chain was annotated to Wikipedia by using Amazon Mechanical Turk. The accuracy of the majority vote on these annotations was 85%. The dataset quality was further improved by manual corrections.

A dataset of around 500 tweets taken from ‘verified accounts’ was gathered by [15]. They were manually annotated and around 100 tweets were marked as erroneous or ambiguous. The remaining tweets were annotated with approximately two links per tweet on average.

Our test collection of choice is the IITB dataset\(^7\). It comprises 103 documents from various domains: sports, entertainment, science and technology, health, etc. It was assembled by the authors of [11] with the goal of exhaustive annotation, even marking token segments as NA – meaning no corresponding entity exists. The dataset is fairly large, with 17,200 detected mentions, out of which about 40% are marked as NA.

\(^7\) Dataset is available at http://www.cse.iitb.ac.in/soumen/doc/CSAW/Annot/
In this chapter we describe two baselines against which we compare our method. One of the methods is a simple local method, that takes independent entity linking decisions. The other method, called the random graph walk baseline, is a current solving approach, that collectively disambiguates all entities appearing in a document.

Let us begin by illustrating the random graph walk baseline.

4.1 Random Graph Walk

We have implemented the method described in [7] with the purpose of comparing its performance to the performance of our method. We use the exact settings described in the paper, including the exact parameter values for \( \lambda \) and for the context window size.

This baseline models the entity linking problem in a graph, that captures information regarding local mention-entity compatibility and semantic relations between entity pairs. Then, a random graph walk method is applied on the graph and from the output scores the global best assignment of entities is computed.

First, we perform the mention detection step, that was previously presented in Section 3.2. In summary, token spans are identified by gathering all n-grams, matching them against the anchor text dictionary and filtering by keyphraseness scores. The mention detection output is a list of mentions \( M = \{m_1, m_2, \ldots m_N\} \) where each mention has a token span \( m\.name \), a set of candidate entities \( m.E \subseteq \mathcal{E} \) and an offset where it occurs in the document \( m\.offset \).

Next, we will discuss the entity disambiguation steps that output an entity resolution for each of the identified mentions. Two steps are carried out:
4. Baseline methods

first a graph is constructed and, second, the random graph walk algorithm is applied on the graph.

4.1.1 Referent Graph

An entity linking graph is constructed, called the Referent Graph, that captures global interdependencies between distinct entity linking decisions and local mention-entity compatibilities.

The Referent Graph is a weighted graph \( G = (V, E) \).

The vertex set \( V \) consist of two types of vertices:

- **Mention vertex**: a node is constructed for each identified mention from the mention detection phase.
- **Entity vertex**: a node is constructed for each entity from the set of candidates of the mentions.

The edge set \( E \) captures two types of information:

1. **Local Mention-to-Entity Compatibility**: an edge is added between each mention and each candidate entity of the mention. The weight of the edge measures how likely the entity is the correct disambiguation given the local context of the mention. This is computed by measuring the overlap of terms occurring both in the local context of the mention and in the Wikipedia article of the entity.

Specifically, the **mention-entity compatibility** is measured by the cosine distance:

\[
CP(m, e) = \frac{\overrightarrow{m} \cdot \overrightarrow{e}}{|\overrightarrow{m}| \cdot |\overrightarrow{e}|} \tag{4.1}
\]

between the mention representation \( \overrightarrow{m} \): the vector of context terms, weighted by tf-idf, appearing in a window of size 50 (words) around the mention and the entity representation \( \overrightarrow{e} \): the vector of terms from the Wikipedia article of the entity, weighted by tf-idf. We use Wikipedia to compute the document frequencies employed by tf-idf scores.

Evidence is propagated from a mention \( m \) to each candidate entity \( e \). The **edge weights** are normalized such that the sum of outgoing edges from a mention node sums up to one:

\[
w(m \rightarrow e) = \frac{CP(m, e)}{\sum_{e \in m,E} CP(m, e)}
\]

In the rare cases when the mention has zero compatibility to all candidate entities, the formula above yields division by zero. We work around this problem by assigning each edge the same weight \( \frac{1}{|m,E|} \).
Note that the evidence propagation is asymmetric: only from mentions towards candidate entities and not the other way around; in other words the edge is directed.

2. **Semantic Relatedness between entities**: an edge is added between each pair of entity vertices that are related and belong to distinct mentions. The measure of relatedness of entities is inspired by the normalized Google distance [2] and was first used in the context of entity linking in [17]. Formally, the relatedness between entities $a$ and $b$ is given by:

$$SR(a, b) = 1 - \frac{\log(\max(|A|, |B|)) - \log(|A \cap B|)}{\log(|W|) - \log(\min(|A|, |B|))}$$  \hspace{1cm} (4.2)

where $A$ and $B$ are the sets of all entities that link to $a$ and $b$ in Wikipedia respectively and $W$ is the set of Wikipedia articles.

By connecting two entities we allow for evidence to be propagated between distinct entity linking decisions. The edge weights are normalized such that the sum of outgoing edges from an entity node sums up to one. An edge between candidate entities $e_i$ and $e_j$ will be assigned the weight:

$$\text{Edge weight}(e_i \rightarrow e_j) = \frac{SR(e_i, e_j)}{\sum_{e \in N_{e_i}} SR(e_i, e)}$$

where we denoted by $N_{e_i}$ the set of entities in the graph that co-occur with entity $e_i$ in at least one article from the Wikipedia corpus.

Note that we only add to the graph edges between entities that have positive relatedness scores.

An example of a referent graph is depicted in Figure 4.1.
4. Baseline methods

4.1.2 Collective entity linking

On the constructed referent graph, a collective entity linking algorithm is applied, that aims to make each entity resolution decision, by taking two criteria into consideration:

- A strong compatible relation between the mention and the entity exists.
- The entity is topically coherent with the other resolved entities of the document.

Initial evidence

Initial evidence accounts for the fact that certain mentions are more informative than other mentions and they should have a larger influence on the entity linking process.

A prior importance score is assigned to each mention \( m \) based on its tf-idf score. Further, the score is normalized such that the importance scores of all mentions in a document \( D \) sum up to one:

\[
\text{Importance}(m) = \frac{\text{tf-idf}(m)}{\sum_{m \in D} \text{tf-idf}(m)}
\]

The Random Graph Walk algorithm

The collective entity linking algorithm used is the Random Graph Walk with Restart algorithm [26], which is equivalent to the Personalized PageRank algorithm [8].

The algorithm is applied on the Referent Graph. Let us assume the graph has \( n \) nodes indexed from 1 to \(|V|\) that are either mention nodes or entity nodes. We use the following notations:

\( s \): The initial evidence vector of \( n \times 1 \) dimensions, with \( s_i = \text{Importance}(m) \), if node \( i \) is a mention node and \( s_i = 0 \) if \( i \) is an entity node.

\( r \): The vector of PageRank scores for the graph nodes, computed by the algorithm in rounds, with \( n \times 1 \) dimensions where \( r_i \) is the score of node \( i \).

\( T \): The transition matrix of \( n \times n \) dimensions with \( T_{ij} \) being the weight of the edge from \( j \) to \( i \). Missing edges are assigned a zero weight.

The vector \( r \) is updated in rounds:

1. Initially \( r^0 = s \).
2. Until the convergence criterion $|r^{t+1} - r^t| < \tau$ is fulfilled, apply:

$$r^{t+1} = (1 - \lambda) \times T \times r^t + \lambda \times s$$

The parameter $\lambda$ is the restart probability and is set, as in the paper, to $\lambda = 0.1$. We set the parameter $\tau = 10^{-5}$.

Note that because we use $\lambda \neq 0$, the algorithm is guaranteed to converge.

A closed form solution exists:

$$r = \lambda \cdot (I - (1 - \lambda) \cdot T)^{-1} \cdot s$$

where $I$ is the identity matrix. Yet, in practice we do not use the close form solution, but rather apply the updates until convergence. The reason is that the matrix inversion is hard to compute accurately and in reasonable time for large matrices.

The final solution picks the best entity assignment $e$ for each mention $m$ that satisfies:

$$e = \arg \max_{e \in m,E} CP(m, e) \times r_d(e)$$

where $r_d(e)$ is the final PageRank score of the vertex corresponding to entity $e$.

**Implementation details**

We use the JUNG library\(^1\) that provides an implementation of the Personalized PageRank algorithm, based on the paper [8].

In order to compute local compatibility between mentions and entities, we need to precompute document frequencies of terms in Wikipedia. Furthermore, for each entity, we want to precompute a sparse vector of tf-idf weights for each term occurring in the entity’s article.

For a scalable, fast implementation, we build an index of document frequencies, for all terms in Wikipedia, using a Map-Reduce solution. We then construct an index of tf-idf term vectors for each Wikipedia canonical entity. While all other indices are stored in memory, we store the tf-idf entity index on disk, because of its large size. We split the index into ten sequence files. Then, we construct a separate index that maps each entity to the location of it’s tf-idf term vector: the file and the offset in the file where the entry occurs. The second index has a small size and can be kept in memory. At query time, the file and offset of the entity entry are retrieved from the second index and a single disk seek is made to retrieve the tf-idf term vector from the first index.

The semantic relatedness measure can easily be computed by querying the entity links index. The construction of this index was detailed in 3.1.1.

\(^1\)http://jung.sourceforge.net/
4. Baseline methods

4.2 Most Frequent Entity

Our second baseline is a trivial local method that makes independent entity linking decisions. It is frequently used for comparison by many entity linking papers.

The mention detection step identifies all mentions that will be linked in an input document: a list of mentions $M = \{m_1, m_2, \ldots, m_N\}$ where each mention has a token span $m.name$, a set of candidate entities $m.E \subseteq E$ and an offset where it occurs $m.offset$. This step was presented in Section 3.2.

The next step is, for each mention $m$, to choose the entity $e$ most frequently linked: $e = \arg \max_{e \in m.E} P(e|m.name)$.

We precompute the probabilities $P(e|m.name)$ via a map-reduce solution for all names and candidate entities from the anchor text dictionary. We estimate these probabilities by counting over the Wikipedia corpus:

$$P(e|n) = \frac{L(n, e)}{\sum_{e' \in E} L(n, e')}$$

where $L(n, e)$ is the number of Wikipedia links with anchor text $n$ and target entity $e$. 
In this chapter, we present our approach for entity disambiguation: using loopy belief propagation to solve the maximum a posteriori (MAP) problem that identifies the most likely assignment of entities for the observed mentions.

In the first section, we describe the underlying graphical model that encodes our assumptions regarding the structure of the problem. We outline in the second section the inference algorithm that we use. In the last section we discuss how we model the clique potentials of the joint distribution and how we calibrated them over observed data.

5.1 The underlying graphical model

The uncertainty in the entity linking problem can be captured by relevant random variables and the interactions between them. Given a document, the first step is to perform mention detection as presented in Section 3.2 to identify mentions of entities. Then, given this evidence, two types of variables are relevant: observed random variables – the names of mentions identified in the mention detection step and hidden random variables – the underlying entities that we want to discover. We will call the variables in the first category: mention variables and the ones in the second category: entity variables.

These random variables exhibit certain relations: for instance, we assume that a mention variable only depends on its underlying entity and that a mention variable is independent on the other entity variables, if the state of its underlying entity is known. Furthermore, relations between co-occurring entities exist: semantically related entities are more likely to appear together in a document. We want to construct a model that can capture these relations and exploit them to find the best assignment of entities for the hidden
Probabilistic graphical models have become extremely popular tools, useful for representing joint probability distributions over a set of random variables and for modelling dependence and independence properties between these variables by using graphs. The advantage of framing the entity linking problem in a graphical model setting is that standard algorithms can be used for inference. For example, belief propagation techniques can solve two types of problems. First, the sum-product algorithm can compute marginal probabilities given observed variables. Second, the max-sum algorithm, also known as max-product or min-sum, can be used to compute the most likely state of the variables given observed evidence. Depending on the underlying dependencies in the graphical model, inference can be either exact or approximate. For instance, approximate inference with loopy belief propagation has been shown to work well on certain applications [20].

Graphical models cast the problem in a graph setting that incorporates the dependency structure between random variables. Each node in the graph represents a random variable (or a group of random variables) and the edges between the nodes in the graph reveal probabilistic relations between the corresponding variables. The overall graph structure expresses the way in which the joint distribution over all of the random variables decomposes into a product of factors, each depending only on a subset of these variables.

The two most common types of graphical models are Bayesian networks, also known as directed graphical models, and Markov random fields, also known as undirected graphical models. The difference between the two models is in the types of dependencies between random variables that can be expressed. The former models are represented through directed acyclic graphs and capture conditional dependencies among variables, while the latter are represented by undirected graphs and can model relations between variables where one cannot naturally assign a directionality to the interaction. In our case, relations between co-occurring entities are best described by an undirected interaction and in consequence we use Markov random fields to model the structure of the entity linking problem.

A Markov random field is represented by an undirected graph $G = (V, E)$ that accompanies a set of random variables $X = (X_v)_{v \in V}$ following a specific
5.1. The underlying graphical model

distribution.

An associated parametrization specifies how the joint distribution factorizes.
In order to specify the parametrization we must first define a few concepts.

**Definition 5.1** A **clique** is a subset of the nodes in the graph, such that every pair of nodes in the subset are connected by a link.

**Definition 5.2** A **maximal clique** is a clique that cannot be extended with additional nodes, without it ceasing to be a clique.

These concepts are depicted by an undirected graphical model over four variables in Figure 5.1. The graph outlines three cliques: \( C_1 = \{X, Y\} \), \( C_2 = \{Y, Z, T\} \) and \( C_3 = \{Z, T\} \). Both \( C_1 \) and \( C_2 \) are maximal cliques. However, the set of nodes \( \{X, Y, Z, T\} \) is not a clique because of the missing links from \( X \) to \( Z \) and from \( X \) to \( T \).

**Definition 5.3** A **potential function** \( \psi(x) \) or a **joint potential function** \( \psi(x_1, x_2, \ldots, x_n) \) is a non-negative real-valued function of its arguments.

Let us denote a full state of the random variables by \( x = (x_v)_{v \in V} \) where each \( x_v \) is the state of the random variable \( X_v \). Let \( C \) be the set of all cliques in the graph and let \( C \) denote a specific clique. We use the notation \( x_C = (x_v)_{v \in C} \) for the state of the variables in the clique.

The joint distribution of the random variables in a Markov random field factorizes as a product of potential functions \( \psi_C(x_C) \) over the maximal cliques in the graph:

\[
P(X = x) = \frac{1}{Z} \prod_{C \in C} \psi_C(x_C)
\]

where \( Z \) is a normalization constant, called the **partition function**, with the value:

\[
Z = \sum_x \prod_{C \in C} \psi_C(x_C)
\]

that ensures the normalization condition for the distribution holds:

\[
\sum_x P(X = x) = 1
\]

Since the choice of potential functions is restricted to functions that satisfy \( \psi_C(x_C) \geq 0 \), we can guarantee that \( P(x) \geq 0 \).

For the example in Figure 5.1, the joint distribution can be factorized as:

\[
P(X = x, Y = y, Z = z, T = t) = \frac{1}{Z} \psi_{C_1}(x, y) \cdot \psi_{C_2}(y, z, t).
\]

Although we stated the factorization property over maximal cliques, often simplifying assumptions are made, such as decomposition of potential functions over maximal cliques as a product of potential functions over non-maximal cliques. For the example in Figure 5.1, the joint distribution can be
decomposed in pairwise factors as:

\[
P(X = x, Y = y, Z = z, T = t) = \frac{1}{Z} \psi_{XY}(x, y) \cdot \psi_{YZ}(y, z) \cdot \psi_{YT}(y, t) \cdot \psi_{ZT}(z, t)
\]

**Defining the graphical model for the entity linking problem**

In the case of the entity linking task, we construct a graphical model for each given input document. We want to model two types of relations: mention variables probabilistically interact only with their corresponding entity variable, while entity variables also interact with all other entities appearing in the document. The mention variables are observed, but their underlying entities are hidden. The set of possible states of an entity variable is the set of candidate entities, that were identified in the mention detection step, belonging to its corresponding mention. A graph that expresses these relations has a node for each mention and a node for the underlying entity of each mention. Each mention is linked to its associated entity and each entity is also linked to all other entities in the document. Note that distinct mentions, even if they refer to the same entity, are linked to separate corresponding entity nodes.

A depiction of the underlying graphical model, corresponding to a document with five mentions is shown in Figure 5.2. The set of mention nodes is \{M_1, M_2, M_3, M_4, M_5\}. The fact that these are observed variables is indicated by shading the nodes. The set of entity nodes is \{E_1, E_2, E_3, E_4, E_5\}. All entity nodes are linked with each other and each mention node is linked with its corresponding entity node. In other words, the subgraph of all entity nodes is complete.

![Figure 5.2](image)

*Figure 5.2:* The underlying graphical model of a document with five mentions used for the entity linking task. Observed variables are shaded.

All input documents can be represented by an undirected graphical model in a similar way.
We will further associate a parametrization to the joint distribution: it is easy to model the joint distribution of mentions and entities by restricting its decomposition to pairwise factors. This means that we factorize the joint distribution over cliques of size two: we model potential functions over mention–entity pairs and over entity–entity pairs. Note that univariate potentials can be grouped inside mention–entity potentials. Statistics on co-occurrence of entities and on how frequently entities are mentioned under certain names can be incorporated in these potentials.

Higher order factors are more difficult to model because we would need to collect statistics on co-occurrence of triplets or larger order tuples of entities. Then, the faced problems are data sparsity and memory requirements for storing the large set of relations. The complexity of inference also increases when modelling higher order factors because we need to evaluate each factor for all its possible states.

For these reasons, we restrict the parametrization of our model to pairwise factors. This special case of a graphical model is called a pairwise Markov random field.

The derivation of the inference algorithm that we use requires the factor graph framework when it is applied to graphical models that are parametrized by potential functions of more than two variables. To simplify notation, we only present the inference algorithm in the context of pairwise Markov random fields, such that the factor graph framework is not needed.

5.2 Inference in Pairwise Markov Random Fields

The joint distribution in a pairwise Markov random field decomposes as a product of node potentials $\psi_i(x_i)$ and edge potentials $\psi_{ij}(x_i, x_j)$:

$$P(X = x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \cdot \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j)$$  \hspace{1cm} (5.1)

Note that the notations $\psi_{ij}(x_i, x_j)$ and $\psi_{ji}(x_j, x_i)$ can be used interchangeably.

Inference algorithms for graphical models solve two classes of problems: computing marginals and finding the most likely assignment of a subset of the variables given observed evidence. Finding marginals does not also find the most likely joint assignment. The values of the latent variables that maximize the marginals can be different and have a suboptimal joint probability than the ones found by MAP techniques. Therefore, we present the max-sum algorithm used for MAP inference and not the sum-product algorithm used for computing the marginals.

In the next section, we will present the max-sum algorithm, that can be used for exact inference on trees. The generalization of this algorithm to graphs
5. Loopy Belief Propagation

with cycles is achieved by loopy belief propagation and will be presented afterwards.

5.2.1 The Max-Sum Algorithm

The max-sum algorithm takes as input a tree graph \( G = (V, E) \), that in our case represents a pairwise Markov random field. We use the notation \( V = \{1, 2, \ldots, N\} \) for the vertices of the graph, the notation \( X = (X_i)_{i \in \mathbb{V}} \) for the set of random variables associated with the graph and \( x = (x_i)_{i \in \mathbb{V}} \) for a state of the variables. The factorization of the joint distribution of the random variables follows the typical factorization of a pairwise Markov random field expressed by Equation (5.1).

The max-sum algorithm is an application of dynamic programming that computes the most likely assignment of a set of variables:

\[
x^* = \arg \max_x P(X = x)
\]

The largest value of the joint probability is then given by:

\[
P(x^*) = \max_x P(X = x) = \max_{x_1} \max_{x_2} \cdots \max_{x_N} P(X_1 = x_1, \ldots, X_N = x_N) \tag{5.2}
\]

Before giving the description of the algorithm, let us first give the intuition on an example, the graphical model in Figure 5.3. The distribution of the variables in the graph factorizes according to (5.1):

\[
P(x_1, x_2, x_3, x_4) = \frac{1}{Z} \prod \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \psi_{24}(x_2, x_4) \prod \psi_i(x_i) \tag{5.3}
\]

The key observation is that, when we evaluate the maximum of the joint probability, the products and the maximizations can be interchanged:

\[
\max_x P(X = x) = \frac{1}{Z} \max_{x_1} \cdots \max_{x_4} \psi_{12}(x_1, x_2) \psi_{23}(x_2, x_3) \psi_{24}(x_2, x_4) \prod \psi_i(x_i)
\]

For the last equality we have used the distributive property \( \max(ab, ac) = a \max(b, c) \) that holds if \( a \geq 0 \) (as will always be the case for potential functions).
5.2. Inference in Pairwise Markov Random Fields

We can perform the computation in the last equation from right to left. We can view each intermediate result as a function of one argument. We compute the values of each of these functions for each possible state, as follows:

\[ m_{X_4 \rightarrow X_2}(x_2) = \max_{x_4} \psi_{24}(x_2, x_4) \psi_4(x_4), \quad \forall x_2 \]  
(5.4)

\[ m_{X_3 \rightarrow X_2}(x_2) = \max_{x_3} \psi_{23}(x_2, x_3) \psi_3(x_3), \quad \forall x_2 \]  
(5.5)

\[ m_{X_2 \rightarrow X_1}(x_1) = \max_{x_2} \psi_{12}(x_1, x_2) \psi_2(x_2) m_{X_3 \rightarrow X_2}(x_2) m_{X_4 \rightarrow X_2}(x_2), \quad \forall x_1 \]  
(5.6)

The largest joint probability is then given by:

\[ \max_x P(x) = \max_{x_1} \psi_1(x_1) m_{X_2 \rightarrow X_1}(x_1) \]

To retrieve the assignment of variables \( x^* = (x^*_i)_{i \in \mathcal{X}} \) that maximizes the joint probability, we need to trace back the results. We compute the best assignment as follows:

\[ x_1^* = \arg \max_{x_1} \psi_1(x_1) m_{X_2 \rightarrow X_1}(x_1) \]

\[ x_2^* = \arg \max_{x_2} \psi_{12}(x_1^*, x_2) \psi_2(x_2) m_{X_3 \rightarrow X_2}(x_2) m_{X_4 \rightarrow X_2}(x_2) \]  
(5.7)

\[ x_3^* = \arg \max_{x_3} \psi_{23}(x_2^*, x_3) \psi_3(x_3) \]

\[ x_4^* = \arg \max_{x_4} \psi_{24}(x_2^*, x_4) \psi_4(x_4) \]

In case the assignment is not unique, ties will arise in (5.7). They can be broken randomly to find one of the possible solutions. The other solutions can be found by tracing back all assignment possibilities that result in ties.

Note that we can interpret this algorithm in terms of propagating local messages around the tree. An example is depicted in Figure 5.3. We choose the node of the variable \( X_1 \) as the root. The leaf nodes send messages to their parent with values given by equations (5.4) and (5.5). After receiving these messages, the node of the variable \( X_2 \) uses this information to compute a new message with the value given by equation (5.6) and sends it to the root. Finally we can compute the best assignment by using the group of equations (5.7). These again can be interpreted as passing messages from the root to the leaves. Each node computes the value.

![Figure 5.3: An undirected graphical model with four variables outlining the messages propagated by the max-sum algorithm.](image)
of its variable by applying the corresponding equation and, then, passes it
down to its children.

Next we will show the generalization of this procedure, the max-sum algo-

We fix a node as root. Then, we can define \( Y_s \) to be the set of all variables in
the subtree that has node \( X_s \) at the root, for each \( X_s \). Let \( X_p \) be the parent of
node \( X_s \) and \( x_p \) one of its states. We want the message \( m_{X_s \rightarrow X_p}(x_p) \) to have
the value of the maximization over all states of the variables in \( Y_s \) of the
product of the potential functions:

\[
\psi_i(x_i), \psi_{ij}(x_i, x_j), \psi_{sp}(x_s, x_p), \quad \forall i : X_i \in Y_s, \forall j : j > i, X_j \in Y_s.
\]

Note that these are all potential functions that take at
least one variable from \( Y_s \) as an argument.

For the example graph in Figure 5.3, we have \( Y_2 = \{ X_2, X_3, X_4 \} \) and \( Y_3 = \{ X_3 \} \). We can then write:

\[
m_{X_2 \rightarrow X_1}(x_1) = \max_{x_2, x_3, x_4} \psi_2(x_2) \psi_3(x_3) \psi_4(x_4) \psi_{23}(x_2, x_3) \psi_{34}(x_2, x_4) \psi_{12}(x_1, x_2)
\]

\[
m_{X_3 \rightarrow X_2}(x_2) = \max_{x_3} \psi_3(x_3) \psi_{23}(x_2, x_3)
\]

The general definition of the messages is easy to be given recursively:

\[
m_{X_s \rightarrow X_p}(x_p) = \max_{x_s} \psi_{ps}(x_p, x_s) \psi_s(x_s) \prod_{c \in c(s)} m_{X_c \rightarrow X_s}(x_s) \quad (5.8)
\]

We use the notation \( c(i) \) for the set of children nodes of node \( i \).

In case node \( X_s \) is a leaf node it does not receive any messages and we can
consider that \( \prod_{c \in c(s)} m_{X_c \rightarrow X_s}(x_s) = 1 \) in the formula above.

The algorithm works by propagating messages from the leaves towards the
root. Each node \( X_s \) sends a message to its parent node \( X_p \) for each possible
value \( x_p \) of the parent node. The value of the messages is given by Equation
(5.8).

Let \( X_r \) be the root node and \( x_r \) one of its states. We can now write the
maximization of the joint distribution in terms of messages propagated from
the root’s children towards the root:

\[
\max_x P(x) = \max_{x_r} \psi_r(x_r) \prod_{c \in c(r)} m_{X_c \rightarrow X_r}(x_r) \quad (5.9)
\]

Then, the best entities assignment is found by tracing back the variables
chosen in the maximization operations. The optimal outcomes are computed
in a top-down approach starting from the root towards the leaves. Each
node computes the best value of its variable and, then, passes it down to its
5.2. Inference in Pairwise Markov Random Fields

children nodes.

\[
x^*_r = \arg \max_{x_r} \psi_r(x_r) \prod_{c \in \langle r \rangle} m_{X_c \rightarrow X_r}(x_r)
\]

\[
x^*_n = \arg \max_{x_n} \psi_{np}(x_n, x^*_p) \psi_n(x_n) \prod_{c \in \langle n \rangle} m_{X_c \rightarrow X_n}(x_n), \quad \forall n \in V \setminus \{r\} \tag{5.10}
\]

where we denoted by \( p \) the parent node of node \( n \). Note that we store all the messages values \( m_{X_c \rightarrow X_p}(x_p) \) after they are computed, such that we do not compute the value of a message more than once.

Because the product of many small values of messages can lead to underflow, it is convenient, in practice, to work with the logarithm of the messages. The effect of this change is that products are replaced by sums. Equation (5.8) becomes:

\[
\ln m_{X_c \rightarrow X_r}(x_r) = \max_{x_s} \ln \psi_{rs}(x_r, x_s) + \ln \psi_s(x_s) + \sum_{c \in \langle s \rangle} \ln m_{X_c \rightarrow X_s}(x_s) \tag{5.11}
\]

Since the logarithm is a monotonic function, the maximum function and the logarithm function can be interchanged. Therefore, we can compute the maximum of the joint probability from the logarithm of the messages by adapting (5.9):

\[
\max_x P(x) = \max_x \ln P(x) = \max_{x_r} \ln \psi_r(x_r) + \sum_{c \in \langle r \rangle} \ln m_{X_c \rightarrow X_r}(x_r)
\]

Finally, we can find the best assignment in terms of the logarithm of the messages by adapting (5.10):

\[
x^*_r = \arg \max_{x_r} \ln \psi_r(x_r) + \sum_{c \in \langle r \rangle} \ln m_{X_c \rightarrow X_r}(x_r)
\]

\[
x^*_n = \arg \max_{x_n} \ln \psi_{np}(x_n, x^*_p) + \ln \psi_n(x_n) + \sum_{c \in \langle n \rangle} \ln m_{X_c \rightarrow X_n}(x_n) \tag{5.12}
\]

where the optimal value of the the parent node of the node corresponding to \( X_n \) is denoted by \( x^*_p \) and \( x^*_r \) is the optimal value of the root node.

Following the computation of messages from (5.11) and the computation of the best assignment values from (5.12) leads to the max-sum algorithm. Sometimes, the negative logarithm of the messages is used and minimization replaces maximization. In this case the algorithm is called min-sum.

To analyse the efficiency of the algorithm, let us suppose the input consists of \( N \) discrete random variables, each with \( K \) possible states. One message \( m_{X_c \rightarrow X_p} \) retains \( K \) values, that each require \( K \) maximization steps to be computed, and in each maximization step we make \( O(\text{degree}(X_c)) \) operations.
5. Loopy Belief Propagation

Since a message is sent over each edge and there are \( N - 1 \) edges, the overall complexity is polynomial \( \sum_{(i,j) \in E} O(K^2 \cdot \text{degree}(i)) = O(K^2 \cdot N) \). In contrast, a naive solution that evaluates the joint probability for all possible state combinations has an exponential complexity \( O(K^N) \).

It is straightforward to include observed evidence in this algorithm. While for the hidden variables, we maximize over all possible states, for the observed values, we maximize over the single observed state, effectively removing the maximization operator.

In the case of the entity linking problem, we observe the mention variables \( M = \{M_1, M_2, \ldots, M_N\} \) and we want to find the state of the hidden entity variables \( E = \{E_1, E_2, \ldots, E_N\} \). Let \( m = (m_i)_{i \in \mathbb{N}} \) be the observed names of the mentions and \( e = (e_i)_{i \in \mathbb{N}} \) be a possible state of the entities. We want to find the best assignment of the entity variables that maximizes the posterior probability:

\[
e^* = \arg \max_e P(E = e | M = m)
\]

Since the mention variables are observed, maximizing the posterior is equivalent to maximizing the joint probability, by applying Bayes rule:

\[
\arg \max_e P(E = e | M = m) = \arg \max_e \frac{P(E = e, M = m)}{P(M = m)} = \arg \max_e P(E = e, M = m)
\]

As previously mentioned, we assume the joint distribution of mentions and entities factorizes in pairwise and univariate factors as in (5.1). Thus, we can apply the max-sum algorithm that we have just presented when the underlying graphical model is a tree. This happens rarely, only for documents with a maximum of two mentions; for all other documents the entity nodes form a clique with at least one cycle in the graph. In the general case, we apply an extension of the max-sum algorithm to graphs with loops, that we present in the next section.

5.2.2 Loopy belief propagation

A simple approach to solving the MAP problem in general graphs is to apply the max-sum algorithm despite the fact that the algorithm may not converge or may not give the correct results. This is a variant of the loopy belief propagation method that has been shown to perform well on certain tasks, for instance by [20].

Since the graph now has cycles, information flows many times around the graph until the values of the messages converge. In order to apply this approach, we need to specify a message passing schedule. We use the standard
synchronous schedule that updates the messages in rounds based on the messages received in the previous round.

During a round, messages are passed over all edges of the graph in both directions. Intuitively, the message \( m_{X_i \rightarrow X_j}(x_j) \) represents the belief of node \( X_i \) that node \( X_j \) should be in state \( x_j \). The value of this message depends on all messages received in the previous round, except the message received from \( X_j \). The message update rule is:

\[
m^{t+1}_{X_i \rightarrow X_j}(x_j) = \max_{x_i} \ln \psi_i(x_i) + \ln \psi_{ij}(x_i, x_j) + \sum_{k \in ne(i) \setminus \{j\}} m^t_{X_k \rightarrow X_i}(x_i)
\] (5.14)

where we denote by \( t \) the number of the round in which the message is transmitted and by \( ne(i) \) the set of neighbour nodes of node \( i \).

As we have previously discussed, we work with the logarithm of the joint distribution to avoid underflow.

The problem of message initialization arises, which we solve by initializing all messages to 0:

\[
m^0_{X_i \rightarrow X_j}(x_j) = 0
\] (5.15)

We test for convergence by analysing the change in the values of the messages at the end of a round. When this change is below a small threshold we stop the message passing. The stopping condition is:

\[
\max_{(i,j) \in E \setminus i} \left| m^{t+1}_{X_i \rightarrow X_j}(x_j) - m^t_{X_i \rightarrow X_j}(x_j) \right| \leq \tau
\] (5.16)

where we have set \( \tau = 10^{-5} \).

After convergence we compute the belief values of each node:

\[
b_i(x_i) = \ln \psi(x_i) + \sum_{j \in ne(i)} m_{X_j \rightarrow X_i}(x_i)
\] (5.17)

The MAP assignment of variables is then given by:

\[
x_i^* = \arg \max_{x_i} b_i(x_i)
\] (5.18)

When working in the log domain, messages can be positive or negative. To avoid overflow and underflow, we perform message normalization in each round. Note that adding a constant to the values of the message \( m_{X_i \rightarrow X_j}(x_j), \forall x_j \) does not change the result of the algorithm. We uniformly shift all values of a message such that the lowest value is 0:

\[
m_{X_i \rightarrow X_j}(x_j)^{\text{norm}} = m_{X_i \rightarrow X_j}(x_j) - \min_{x_j} m_{X_i \rightarrow X_j}(x_j)
\] (5.19)
At this point, it is worth giving a summary of the algorithm. We start by initializing messages as in (5.15). Until convergence, messages are sent along all edges in both directions in successive rounds. In each round, the unnormalized messages are computed as in (5.14) and, then, normalized by applying (5.19). We stop sending messages when the convergence criterion in (5.16) is achieved. After convergence, the belief values are computed for each node and each of its states as in (5.17). The solution is then found by applying (5.18).

Let us work out the computation cost of the loopy belief propagation algorithm. Let us assume the input consists of \( N \) discrete random variables organized in a graph with \( E \) edges that each can take \( K \) values. The cost of computing a message \( m_{X_i \rightarrow X_j} \) is \( O(K^2 \cdot \text{degree}(X_i)) \) because there are \( K \) values to be computed, each requiring a maximization over \( K \) steps, and each maximization requires a product over \( O(\text{degree}(X_j)) \) values. In each round, two messages are sent over each edge, leading to \( \sum_{(i,j) \in E} O(K^2 \cdot \text{degree}(i)) = O(K^2 \cdot E) \) complexity per round. Let’s denote by \( I \) the number of iterations the algorithm will do until convergence. The final complexity is then \( O(I \cdot E \cdot K^2) \).

### 5.3 Joint probability models for entity linking

Our approach for the entity disambiguation task is to pick the assignment of entities that maximizes the posterior probability. Previously, we have shown in (5.13) that maximizing the posterior \( P(E = e | M = m) \) is equivalent to maximizing the joint probability \( P(E = e, M = m) \) when the mention variables are observed. We have modelled the underlying entity linking graph in Section 5.1 and we have yet to associate a parametrization of the joint distribution of mentions and entities. After this is accomplished, we can apply the loopy belief propagation algorithm to find the most likely assignment of entities.

As we have previously stated, for efficiency reasons, we restrict the factorization of the joint distribution of mentions and entities to clique potentials of at most two variables:

\[
P(E = e, M = m) = \frac{1}{Z} \prod_{1 \leq i \leq N} \psi_i(e_i, m_i) \cdot \prod_{1 \leq i < j \leq N} \psi_{ij}(e_i, e_j) \quad (5.20)
\]

To simplify notation, we have considered that the effect of the observed mentions is absorbed by the univariate potentials \( \psi_i(e_i, m_i) \). Similarly, we have considered the effect of the entity priors \( \psi_i(e_i) \) to be absorbed by the same potentials \( \psi_i(e_i, m_i) \).

So far, we have neglected the choice of potential functions for the entity linking problem. In general, potential functions are not restricted to those
having a specific probabilistic interpretation as marginal or conditional distributions. Nevertheless, they are interpretable: states with higher potential values are more probable.

Our first attempt is to use a basic model for the potential functions inspired by the work in [11]. This model uses as potential functions the exponential of the local compatibility measure between a mention and its context and the exponential of the relatedness measure between entities, that were defined in the random graph walk baseline.

However, this choice of modelling is ‘ad-hoc’. Ideally, the chosen potentials should be calibrated such that they maximize the likelihood of observed data. As we shall see, this is not a simple problem and in practice we make a simplification of the problem that enables us to compute the potentials in polynomial time.

We refer to the previously mentioned ‘ad-hoc’ model as the uncalibrated model and to its modified version as the calibrated model. They are presented in the next two sections.

### 5.3.1 Uncalibrated model

Assuming the joint distribution is strictly positive, we can model it through a log-linear model, as in [11]:

\[
P(E = e, M = m) = \frac{1}{Z} \exp \left[ \sum_{1 \leq i \leq N} CP(e_i, m_i) + \sum_{1 \leq i < j \leq N} SR(e_i, e_j) \right]
\]

where \( CP(e_i, m_i) \) is an entity-mention compatibility function and \( SR(e_i, e_j) \) is a function measuring the semantic relatedness between the entities.

In other words, we define the clique potentials in terms of compatibility and relatedness functions as follows:

- \( \psi_i(e_i, m_i) = \exp \left[ CP(e_i, m_i) \right] \)
- \( \psi_{ij}(e_i, e_j) = \exp \left[ SR(e_i, e_j) \right] \)

For convenience, we can choose the functions used in the random graph walk method to be used in the clique potentials modelled here. That is, for \( CP(e_i, m_i) \) we use the cosine similarity between the context of the mention and the representation of the Wikipedia article of the entity, given by Formula (4.1). Further in the case of the semantic relatedness function \( SR(e_i, e_j) \), we use the function from (4.2), that is the normalized Google distance applied in the context of the entity linking problem.

The message update formula (5.14) for loopy belief propagation can now be rewritten in terms of the compatibility function and the relatedness function...
5. Loopy Belief Propagation

as:

\[ m_{E_i \rightarrow E_j}^{t+1}(e_j) = \max_{e_i} CP(e_i, m_i) + SR(e_i, e_j) + \sum_{k \in ne(i) \setminus \{j\}} m_{E_k \rightarrow E_i}^{t}(e_i) \]

Similarly, we can rewrite the solution assignment equations from (5.18):

\[ e_i^* = \arg \max_{e_i} CP(e_i, m_i) + \sum_{j \in ne(i)} m_{E_j \rightarrow E_i}(x_i) \]

The problem with this model is that we do not know which local compatibility and entity relatedness functions are best to use. Furthermore, the resulting probability model may not be a good fit for the observed data. For instance, from a large entity-linked corpus, we can empirically estimate priors over mention-entity pairs \( P(e_i, m_i) \) and prior probabilities of co-occurrence for entity pairs \( P(e_i, e_j) \). However, these may largely differ from the marginal probabilities of this model.

We try to address these issues with the next model, one that does not impose a specific format of the clique potentials, but rather tries to learn the best potentials from the observations in an entity-linked corpus.

5.3.2 Calibrated model

In order to calibrate the joint probability model, let’s assume a corpus of entity-linked documents is available. We can then extract from the corpus statistics of how often certain entities appear in a document and are referred by a specific name and statistics on how often entity pairs co-occur. Ideally, we would want our joint probability model to be a good fit of the observed data. In other words, we can think of the observed statistics as constraints imposed on the joint probability model.

We follow the maximum entropy principle, that states that if one is to select a probability distribution that satisfies a set of constraints and these constraints do not specify a unique distribution, then it is best to choose the distribution with the largest entropy.

**Maximum entropy models** Let us start with a short description, following the presentation in [13], on the technique of modelling the maximum entropy distribution subject to a set of constraints.

The goal is to estimate an unknown distribution \( P(x) \) of a set \( X \) of discrete random variables, with \( x \) a state of the variables.

Let us suppose that under the true distribution, a set of \( m \) functions, denoted by \( g_i(x) \), \( i \in \overline{1,m} \), average to \( \mu_i \):

\[ \mu_i = \sum_x P(x)g_i(x) \quad \forall i \in \overline{1,m} \]
Assuming that $\mu = (\mu_i)_{i \in 1,m}$ is known, we want to find a distribution $Q(x|\mu)$ that satisfies the constraints given above, that is:

$$
\sum_x Q(x|\mu)g_i(x) = \mu_i \quad \forall i \in 1,m
$$

and at the same time has maximum entropy, where the entropy value is:

$$
H(Q) = -\sum_x Q(x) \log Q(x)
$$

This is a well known optimization problem whose solution is in the exponential family:

$$
Q(x|\mu) = \exp \left[ \sum_{i=1}^m \lambda_i(\mu)g_i(x) \right] Z(\mu)
$$

Usually, the true values of $\mu_i$ are not known and are instead estimated from available data. For instance, if we have $K$ observations of the random variables $X$, denoted by $x^{(k)}$, $k \in 1,K$, then the estimate $\hat{\mu}_i$ of $\mu_i$ is:

$$
\hat{\mu}_i = \frac{1}{K} \sum_{k=1}^K g_i(x^{(k)})
$$

Assuming the number of available observations is large enough, we can then still follow the maximum entropy formulation above: we replace $\mu_i$ with $\hat{\mu}_i$ and obtain the solution $Q(x|\hat{\mu})$ given by the Formula (5.21).

**Joint probability constraints** We now want to formalize a set of constraints for the joint distribution of mentions and entities. Then, we can model the maximum entropy distribution that satisfies these constraints as previously shown.

We are given a set $D$ of entity-linked documents. From each document $d$, we extract an observation $(e^{(d)}, m^{(d)}) = (e_i^{(d)}, m_i^{(d)})_{i \in 1,N_d}$, the set of mentions and entities appearing in the document.

We use the notations:

- $\mathcal{E}$ for the entities universe
- $\text{Names}(e)$ for the set of names that are used to refer to an entity $e$
- $\mathcal{U} = \{(e, m)|e \in \mathcal{E}, m \in \text{Names}(e)\}$ for the universe of possible mention-entity pairs.
- $N_d$ for the number of mentions in document $d$
From each observation, we can extract two types of statistics. The first statistic \( g_{e',m'}(e, m) \) computes how many times in an observed document an entity \( e' \) was mentioned under the name \( m' \):
\[
  g_{e',m'}(e, m) = \sum_i \mathbb{1}[e_i = e'] \cdot \mathbb{1}[m_i = m'], \quad \forall (e', m') \in \mathcal{U}
\]
where we have denoted by \( \mathbb{1}[i = j] \) the indicator function:
\[
  \mathbb{1}[i = j] = \begin{cases} 
    0 & \text{if } i = j \\
    1 & \text{if } i \neq j
  \end{cases}
\]
The second type of statistics \( h_{e',e''}(e, m) \) measures how many times in an observed document an entity pair \((e', e'')\) occurred:
\[
  h_{e',e''}(e, m) = \sum_{i < j} \mathbb{1}[e_i = e'] \cdot \mathbb{1}[e_j = e''], \quad \forall e', e'' \in \mathcal{E}
\]
Now, we can empirically compute the average value of these statistics on the observed documents from our corpus. We denote by \( \hat{\mu}_{e',m'} \) the average value of \( g_{e',m'} \) and by \( \hat{\eta}_{e',e''} \) the average value of \( h_{e',e''} \):
\[
  \hat{\mu}_{e',m'} = \frac{1}{|D|} \sum_{d \in D} g_{e',m'}(e^{(d)}, m^{(d)}) = \frac{1}{|D|} \sum_{d \in D} N_d(e', m') \tag{5.22}
\]
\[
  \hat{\eta}_{e',e''} = \frac{1}{|D|} \sum_{d \in D} h_{e',e''}(e^{(d)}, m^{(d)}) = \frac{1}{|D|} \sum_{d \in D} M_d(e', e'')
\]
where we denote by \( N_d(e, m) \) the number of times entity \( e \) occurs under the name \( m \) in document \( d \) and we denote by \( M_d(e', e'') \) the number of entity pairs in document \( d \) where one of the entities is \( e' \) and the other is \( e'' \). Given these observed statistics, the joint distribution of mentions and entities should satisfy the following equations:
\[
  \sum_{e,m} P(e, m | \hat{\mu}; \hat{\eta}) g_{e',m'}(e, m) = \hat{\mu}_{e',m'} \quad \forall (e', m') \in \mathcal{U}
\]
\[
  \sum_{e,m} P(e, m | \hat{\mu}; \hat{\eta}) h_{e',e''}(e, m) = \hat{\eta}_{e',e''} \quad \forall e', e'' \in \mathcal{E}
\]
Following the result in (5.21), the solution for the joint distribution that satisfies the constraints given above and maximizes entropy, is in the exponential family:
\[
  P(e, m) = \frac{1}{Z} \exp \left[ \sum_{(e', m') \in \mathcal{U}} \rho_{e',m'} \cdot g_{e',m'}(e, m) + \sum_{e', e'' \in \mathcal{E}} \lambda_{e',e''} \cdot h_{e',e''}(e, m) \right]
  = \frac{1}{Z} \exp \left[ \sum_i \rho_{i,m_i} + \sum_{i < j} \lambda_{e_i,e_j} \right]
\tag{5.23}
\]
5.3. Joint probability models for entity linking

**Ideal potentials** In order to derive optimal values for the parameters $\rho_{e_i,m_i}$ and $\lambda_{e_i,e_j}$, we can try to maximize the log-likelihood of the observed data:

$$ (\rho^*, \lambda^*) = \arg \max_{\rho, \lambda} \sum_{d \in D} \left\{ \sum_{1 \leq i \leq N_d} \rho_{e_d^i, m_d^i} + \sum_{1 \leq i < j \leq N_d} \lambda_{e_d^i, e_d^j} - \log Z^d \right\} $$

where $Z^d$ is the partition function of document $d$.

Typically, gradient descent-based methods are used for solving this task. The main challenge is that computing the gradient requires evaluating the log-partition function $\log Z^d$. This requires a sum over all configurations of states. Thus, it is at least as computationally expensive as a naive solution to the MAP problem that evaluates the distribution for all configurations and picks the one with the highest probability.

**Uncoupled model** Ideally, we would like to avoid heavy-weight optimization in deriving a good model, but at the same time still obtain calibrated potential functions. The very simplest idea is to ignore all couplings. The uncoupled model is then:

$$ P(e_1:N, m_1:N) = \prod_{i=1}^{N} P(e_i, m_i) = \prod_{i=1}^{N} \frac{\exp[\rho_{e_i, m_i}]}{\sum_{(e', m') \in \mathcal{U}} \exp[\rho_{e', m'}]} $$

where we have used the notation $\mathcal{U}$ for the universe of possible mention-entity pairs.

Note that the model is not identifiable because we can set $\rho_{e, m} \leftarrow \rho_{e, m} + \text{const}$ without changing the model.

From our document corpus, we compute $N(e_i, m_i)$ – the number of times entity $e_i$ is mentioned by the name $m_i$.

We can set $\rho = (\rho_{e, m})_{e, m \in \mathcal{U}}$ by solving the maximum log-likelihood problem:

$$ \rho^* = \arg \max_{\rho} \mathcal{L}(D, \rho) $$

where the log-likelihood $\mathcal{L}(D, \rho)$ can be computed as:

$$ \mathcal{L}(D, \rho) = \log \prod_{d \in D} \prod_{i=1}^{N_d} \frac{\exp[\rho_{e_d^i, m_d^i}]}{\sum_{(e', m') \in \mathcal{U}} \exp[\rho_{e', m'}]} $$

$$ = \log \prod_{(e, m) \in \mathcal{U}} \left( \frac{\exp[\rho_{e, m}]}{\sum_{(e', m') \in \mathcal{U}} \exp[\rho_{e', m'}]} \right)^{N(e, m)} $$

$$ = \sum_{(e, m) \in \mathcal{U}} N(e, m) \left( \rho_{e, m} - \log \sum_{(e', m') \in \mathcal{U}} \exp[\rho_{e', m'}] \right) $$
Taking the derivative yields:

\[
\frac{\partial L(D, \rho)}{\partial \rho_{e,m}} = N(e, m) - \sum_{(e', m') \in U} N(e', m') \frac{\exp[\rho_{e,m}]}{\sum_{(e', m') \in U} \exp[\rho_{e', m'}]} = N(e, m) - \frac{\exp[\rho_{e,m}]}{\sum_{(e', m') \in U} \exp[\rho_{e', m'}]}
\]

By setting the derivative to 0 we get the equations:

\[
\frac{\partial L(D, \rho)}{\partial \rho_{e,m}} = 0 \iff \frac{\exp[\rho_{e,m}]}{\sum_{(e', m') \in U} \exp[\rho_{e', m'}]} = \frac{N(e, m)}{\sum_{d \in D} N_d}
\]

The solution is unique up to a constant $K$:

\[
\rho_{e,m} = \log N(e, m) - K
\]

We fix the constant $K = \log \sum_{d \in D} N_d$ and arrive at the solution:

\[
\rho_{e,m} = \log \frac{N(e, m)}{\sum_{d \in D} N_d} \quad (5.24)
\]

where we can view the fraction on the right hand side as the empirical estimate of the real probability $P(e, m)$:

\[
\frac{N(e, m)}{\sum_{d \in D} N_d} = \hat{P}(e, m) \approx P(e, m) \quad (5.25)
\]

and thus we can rewrite the assignment of $\rho_{e,m}$ to be:

\[
\rho_{e,m} = \log \hat{P}(e, m) \quad (5.26)
\]

**Modelling pairwise entity couplings** The second step is to fix the estimates $\rho_{e,m}$ as in (5.24) and try to come up with a heuristic way of setting $\lambda_{ij}$. To avoid computationally involved solutions, we make a simplification: we assume that all documents contain at most two entity references. In the case of documents with one entity reference, the previous choice for $\rho_{e,m}$ is optimal.

Let’s denote by $N(e, m, e', m')$ the number of documents that contain references to the entity $e$ with the name $m$ and to the entity $e'$ with the name $m'$ and by $N(e, e')$ the number of documents that reference both entities $e$ and $e'$. 

44
5.3. Joint probability models for entity linking

The optimal values of \( \lambda = (\lambda_{ij})_{i,j \in \mathcal{E}} \) should maximize the log-likelihood of the set \( D \) of observed documents with two entities:

\[
\lambda^* = \arg \max_{\lambda} \mathcal{L}(D, \rho, \lambda)
\]

where the value of the log-likelihood is:

\[
\mathcal{L}(D, \rho, \lambda) = \log \prod_{d \in D} \frac{\exp \left[ \rho_{\lambda, D} + \rho_{\lambda, e} + \lambda_{e, d} \right]}{\sum_{(e_1, m_1), (e_2, m_2) \in U} \exp \left[ \rho_{e_1, m_1} + \rho_{e_2, m_2} + \lambda_{e_1, e_2} \right]}
\]

\[
= \log \prod_{(e_1, m_1), (e_2, m_2) \in U} \left( \frac{\exp \left[ \rho_{e_1, m_1} + \rho_{e_2, m_2} + \lambda_{e_1, e_2} \right]}{\sum_{(e, m), (e', m') \in U} \exp \left[ \rho_{e, m} + \rho_{e', m'} + \lambda_{e, e'} \right]} \right)^{N(e_1, m_1, e_2, m_2)}
\]

\[
= \sum_{(e, m), (e', m') \in U} \exp \left[ \rho_{e, m} + \rho_{e', m'} + \lambda_{e, e'} \right]^{N(e_1, m_1, e_2, m_2)}
\]

The value of the derivative is:

\[
\frac{\partial \mathcal{L}(D, \rho, \lambda)}{\partial \lambda_{e, e'}} = \sum_{m \in \text{Names}(e), m' \in \text{Names}(e')} N(e, m, e', m') - \sum_{(e_1, m_1), (e_2, m_2) \in U} \exp \left[ \rho_{e, m} + \rho_{e', m'} + \lambda_{e, e'} \right]^{N(e_1, m_1, e_2, m_2)}
\]

\[
= \sum_{m \in \text{Names}(e), m' \in \text{Names}(e')} \hat{P}(e, m) \cdot \hat{P}(e', m') \cdot \exp[\lambda_{e, e'}]
\]

where

\[
\hat{P}(e) = \sum_{m \in \text{Names}(e)} \hat{P}(e, m)
\]

is the empirical estimate of \( P(e) \).

Setting the derivative to 0, we obtain:

\[
\frac{\partial \mathcal{L}(D, \rho, \lambda)}{\partial \lambda_{e, e'}} = 0 \iff \sum_{(e_1, m_1), (e_2, m_2) \in U} \frac{\exp[\lambda_{e, e'}]}{\exp[\rho_{e_1, m_1} + \rho_{e_2, m_2} + \lambda_{e_1, e_2}]} = \frac{\hat{P}(e_1, e_2)}{\hat{P}(e_1) \cdot \hat{P}(e_2)}
\]
5. Loopy Belief Propagation

where

\[ \hat{P}(e, e') = \frac{N(e, e')}{|D|} \]  

(5.28)

The solution is unique up to a constant \( K \):

\[ \lambda_{e, e'} = \log \left[ \frac{\hat{P}(e_1, e_2)}{\hat{P}(e_1) \cdot \hat{P}(e_2)} \right] - K \]

We set \( K = 0 \) and arrive at the solution:

\[ \lambda_{e, e'} = \log \left[ \frac{\hat{P}(e_1, e_2)}{\hat{P}(e_1) \cdot \hat{P}(e_2)} \right] \]  

(5.29)

Summing up the results, we have computed optimal values for \( \lambda_{e, e'} \) as in (5.29) and for \( \rho_{e, m} \) as in (5.26) by calibrating on observed data. To simplify computation, we have assumed either an uncoupled model, or a set of documents with at most two references of entities. We now want to make use of these values in a model that works for arbitrarily long documents.

**Calibrated joint distribution model**  We can now simply plug-in the computed parameters in the maximum entropy model from (5.23). Using this model, however, does come with challenges. Let \( l \) be the number of mentions in a document. As \( l \) grows, the contributions coming from the pairwise interactions scale with \( (l^2) \), while the contributions coming from local compatibilities scale with \( l \). The simplest approach to counteract this effect is to scale the contributions:

\[ P(e, m) = \frac{1}{Z} \exp \left[ \sum_i \frac{1}{l} \rho_{e, m_i} + \sum_{i<j} \frac{2}{l \cdot (l-1)} \lambda_{e_i, e_j} \right] \]  

(5.30)

This change has a positive impact on the results, which we will present in Table 6.2.

In our model, we use contributions from all entity pairs. However, some entity pairs will be anti-correlated. We do not want to use their contributions, because anti-correlations are not very informative. We would want that the entity linking process would guide itself only on positive correlations of entities.

For this reason, we set the \( \lambda_{e, e'} \) values to 0 when the entities \( e \) and \( e' \) are anti-correlated or independent. Thus, their values are:

\[ \lambda_{e, e'} = \begin{cases} 
0 & \text{if } \hat{P}(e, e') \leq \hat{P}(e) \cdot \hat{P}(e') \\
\log \frac{\hat{P}(e, e')}{\hat{P}(e) \cdot \hat{P}(e')} & \text{otherwise}
\end{cases} \]  

(5.31)
5.3. Joint probability models for entity linking

Note that in the previous formulation \( \lambda_{e,e'} \) cannot be computed when \( \hat{P}(e,e') = 0 \). This new formulation overcomes this problem.

We will compute the needed statistics \( \hat{P}(e,m) \), \( \hat{P}(e) \) and \( \hat{P}(e,e') \) from the Wikipedia corpus with a Map-Reduce solution. Their values are given by Formulas (5.25), (5.27) and (5.28).

To leave some room for errors induced by assumptions and model mismatches, we propose to introduce one tuning parameter \( \beta \) to scale the parameters \( \lambda_{e,e'} \):

\[
P(e,m) = \frac{1}{Z} \exp \left[ \sum_{1 \leq i \leq l} \frac{1}{l} \rho_{e_i,m_i} + \sum_{1 \leq i < j \leq l} \frac{2\beta}{(l-1)} \lambda_{e_i,e_j} \right]
\]

The effect of this change is a slight increase in the training results. The performance impact is presented in the next chapter in Table 6.3.
In this chapter, we present the results of our method compared to the results of our two baselines. We decouple the two entity linking stages: mention detection and entity disambiguation, such that we can analyse their performance separately.

In the first section, we describe the ground truth dataset that we use, called the IITB dataset. We report the evaluation metrics that we compute in the second section. The third section presents the results of each entity linking phase and the overall results.

6.1 IITB Dataset

We compare our results against a ground truth collection, that comprises a set of documents with associated annotations. An annotation specifies an identified mention name, the document and the offset in the document at which the mention occurs, as well as the referred entity.

The testing collection that we use, called the IITB dataset, was collected by the authors of [11]. It comprises 103 documents crawled from popular sites, that belong to various topics including: sports, entertainment, science, technology and health.

The annotations were collected by six volunteers, that were told to annotate exhaustively, even by marking token spans as NA, meaning that no appropriate entity was found in Wikipedia. Our system also aims to annotate exhaustively, but does not mark any token spans as NA. In consequence, we discard these annotations from the dataset.

Moreover, we observed that around 1,500 annotations were linking token spans to disambiguation pages. We do not consider disambiguation pages to be canonical entities, because they do not refer to a specific entity, but
rather a list of possible meanings. As a result, we discard these annotations from the ground truth collection.

After removing the previously mentioned annotations, and after counting distinct spots, around 9,700 annotations remain. This is the set of annotations against which we compare.

6.2 Evaluation metrics

We quantify the performance of our entity linking system by measuring common metrics of success: precision, recall and $F_1$ score.

For a given document, let $M^*$ be the ground truth annotations made for the document, and let $M$ be the output annotations of an entity linking system on the input document. Then our performance metrics are computed as:

- Precision: $P = \frac{|M \cap M^*|}{|M|}$
- Recall: $R = \frac{|M \cap M^*|}{|M^*|}$
- $F_1$ score: $F_1 = \frac{2 \cdot P \cdot R}{P + R}$

In the formulas above, two annotations are considered equal if their names and their annotated entities are the same. Note that the offsets of the annotations are not compared, in order to account for a flaw in the ground truth dataset. If an entity is mentioned multiple times under the same name in a document, the annotations frequently do not cover all mentions, but just a subset of them.

We report the results at macro level. The reported values for precision, recall and $F_1$ scores are the averaged values of the precision, recall and $F_1$ scores over the set of documents.

6.3 Experimental results

In this section, we analyse the performance of our method and compare it with the performance of our two baselines. Since our main improvements are in the entity disambiguation step, it makes sense to decouple the two phases and analyse performance separately. We also present the overall performance of our system.

6.3.1 Mention Detection Performance

The performance of the mention detection step sets an upper bound on the final performance of our entity linking system.
6.3. Experimental results

Let us consider an annotation from the ground truth dataset. If its name does not appear in the name entries of the anchor text index, the name will certainly not be linked by our entity linking system. Likewise, a name will not be linked if it appears in the index, but the correct entity does not occur in its candidates list from the index. This lowers the maximum achievable recall to 96.23%.

When we presented the anchor text index construction, we discussed pruning infrequent candidate entities of a name, that appear in less than 2% of the hyperlinks with the anchor text equal with the name. We will later see that this decision has a positive impact on the final results and on the computation time. However, because candidate entities are pruned from the anchor text index and some of these candidates are annotated in the testing data, the maximum achievable recall turns into 92.88%.

Now, assuming that entity disambiguations runs flawlessly, what are the best results we can achieve, by using the pruned anchor text index? Let us fix a keyphraseness threshold and let us analyse the performance on a single document.

Let $M$ be the set of mentions identified by the mention detection step. Each mention $m$ from the set has $m.name$ and $m.E \subseteq \mathcal{E}$ – a list of candidate entities from the anchor text dictionary of Wikipedia. Mentions with the same name, occurring at distinct offsets are considered only once in the set.

Let $G$ be the set of ground truth annotations of the given document, where an annotation $g$ is specified by $g.name$, $g.entity$.

In the entity disambiguation step, there are two cases for a mention $m$:

- There exists an entity $e$ in $m.E$ such that, in the ground truth dataset, there exists an annotation $g$ with $g.entity = e$ and $g.name = m.name$. In this case the disambiguation may or may not disambiguate correctly. We assume it does disambiguate correctly.

- Otherwise, whatever the outcome of the disambiguation, no matching annotation exists in the ground truth dataset. This can happen because the mention has no corresponding entity in Wikipedia, the token span boundaries are wrong, or the coverage in the ground truth dataset is not perfect. All mentions in this category are false positives and we denote their set by $FP$.

Similarly, for each annotation $g$ in the ground truth set $G$, there are two possible cases:

- There exists a mention $m$, such that $m.name = g.name$ and there exists an entity $e \in m.E$, such that $e = g.entity$. In this case, we have found a true positive. We use the notation $TP$ to denote the set that gathers these annotations.
6. Experiments and results

- Otherwise, this annotation was not found in the mention detection step. This can happen because $g\text{.name}$ had a low keyphraseness score and it was filtered out, or, $g\text{.name}$ is not in the anchor text dictionary, or $g\text{.name}$ is in the anchor text dictionary but does not have a candidate entity $g\text{.entity}$.

Maximum achievable precision and recall can be computed as follows:

$$P = \frac{|TP|}{|TP| + |FP|} \quad \text{and} \quad R = \frac{|TP|}{|G|}$$

(6.1)

Note that after the entity disambiguation stage, the number of linked mentions may be higher than $|M|$, because a mention $m$ in $M$ appearing at multiple offsets can be linked to different entities depending on the local context. This can happen for the random graph walk method and cannot happen for the loopy belief propagation method, as it does not use the local context.

Now we can compute the scores from (6.1), for each document in the test collection and report the final averaged scores. We can vary the keyphraseness threshold and obtain pairs of precision-recall scores. These values represent upper bounds for the overall performance of our entity linking system at the specified keyphraseness levels.

![Figure 6.1: Upper-bound on the results, after mention detection step, depending on keyphraseness threshold.](image)

Figure 6.1 shows the maximum achievable precision, recall and $F_1$ score values, after the mention detection step, depending on the chosen keyphraseness threshold. The same upper bounds, represented in a precision-recall graph, are depicted in Figure 6.2.
6.3. Experimental results

Figure 6.2: Upper-bound on the results, after mention detection step.

6.3.2 Entity Disambiguation Performance

Our main contributions are in the entity disambiguation step. To analyse their performance, we assume the previous mention detection phase ran perfectly. The input for the entity disambiguation is a set of names, taken from the ground truth collection annotations, with a set of candidate entities taken from the anchor text index.

Let us start by analysing the effect of pruning candidates from the anchor text index. Specifically, the candidates that appear in less than 2% of the hyperlinks of a given name, are discarded from the names entry in the anchor text index.

We compare the results of our random graph walk baseline running on the complete anchor text index versus on the pruned anchor text index.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F₁ score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Graph Walk</td>
<td>62.91%</td>
<td>62.00%</td>
<td>62.45%</td>
</tr>
<tr>
<td>Loopy Belief Propagation</td>
<td>83.78%</td>
<td>78.04%</td>
<td>80.91%</td>
</tr>
<tr>
<td>Maximum Achievable</td>
<td>100%</td>
<td>94%</td>
<td>97%</td>
</tr>
<tr>
<td>Random Graph Walk Pruned</td>
<td>82.45%</td>
<td>74.17%</td>
<td>78.31%</td>
</tr>
<tr>
<td>Loopy Belief Propagation Pruned</td>
<td>88.54%</td>
<td>78.45%</td>
<td>83.49%</td>
</tr>
<tr>
<td>Maximum Achievable Pruned</td>
<td>100%</td>
<td>89.38%</td>
<td>94.69%</td>
</tr>
</tbody>
</table>

We compare the results of our random graph walk baseline running on the complete anchor text index versus on the pruned anchor text index.
Likewise, we compare the results of our calibrated model for loopy belief propagation on both anchor text indices. Note that we use the joint distribution given by Formula (5.30) with the $\lambda_{e_i,e_j}$ parameters computed as in Formula (5.31) and the $\rho_{e,m}$ parameters given by (5.26).

The results are shown in Table 6.1 and indicate a major increase in performance. For this reason, from now on, we will always use the pruned index in our experiments. Because some of the ground truth annotations do not occur in the anchor text index, we cannot select a set of candidate entities for them and we do not attempt to disambiguate them. Therefore, the maximum recall is not 100%. By pruning the candidate entities, the maximum recall gets lower, from 94% to around 89%.

We mention that we have experimented with limiting the maximum number of candidates per name to a fixed number: 25 or 100. However the results were lower than those obtained by cutting-off at 2% frequency.

Let us now analyse the performance of the various models for loopy belief propagation that were presented in Section 5.3. The results are summarized in Table 6.2. The model variants that we have tested are:

- **Uncalibrated model**: presented in section 5.3.1.

- **Calibrated unscaled model**: the joint distribution is the maximum entropy distribution from the formula (5.23), where the parameters $\lambda_{ij}$ are specified by formula (5.31) and the $\rho_{ij}$ parameters are given in equation (5.26).

- **Calibrated scaled model**: the joint distribution is illustrated in (5.30) and the parameters are the same as in the calibrated unscaled model.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F₁ score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP Calibrated Unscaled</td>
<td>78.20%</td>
<td>69.30%</td>
<td>73.75%</td>
</tr>
<tr>
<td>LBP Uncalibrated</td>
<td>79.21%</td>
<td>70.23%</td>
<td>74.72%</td>
</tr>
<tr>
<td>LBP Calibrated Scaled</td>
<td>88.54%</td>
<td>78.45%</td>
<td>83.49%</td>
</tr>
<tr>
<td>Maximum Achievable</td>
<td>100%</td>
<td>89.38%</td>
<td>94.69%</td>
</tr>
</tbody>
</table>

We can see that, for arbitrarily long documents, scaling the contributions of the parameters is essential for good results.

We have introduced the $\beta$ parameters in our joint distribution factorization as in 5.32, to account for model mismatches. The training results are shown in Table 6.3. We can see that not using $\beta$ (equivalent to $\beta = 1$) has close
results to the best setting $\beta = 0.8$. We can view this as an indicator that our intuition for scaling the parameters was correct.

Table 6.3: Loopy Belief Propagation Results with $\beta$ parameter

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$ score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>84.92%</td>
<td>75.20%</td>
<td>80.06%</td>
</tr>
<tr>
<td>0.4</td>
<td>87.11%</td>
<td>77.14%</td>
<td>82.12%</td>
</tr>
<tr>
<td>0.8</td>
<td>88.75%</td>
<td>78.62%</td>
<td>83.68%</td>
</tr>
<tr>
<td>1</td>
<td>88.54%</td>
<td>78.45%</td>
<td>83.49%</td>
</tr>
<tr>
<td>2</td>
<td>86.79%</td>
<td>76.90%</td>
<td>81.84%</td>
</tr>
<tr>
<td>10</td>
<td>81.07%</td>
<td>71.87%</td>
<td>76.47%</td>
</tr>
</tbody>
</table>

Let us now present the main results of our method – loopy belief propagation with a calibrated scaled joint distribution, compared with our two baselines in Table 6.4. Our method clearly outperforms both baselines. By modelling the dependencies between the entities that appear together in the text, our method achieves 4-5% higher results than the simple, yet efficient, most frequent entity baseline. Surprisingly, the random graph walk method does not perform better than the other baseline. Also, as show in the table, we can see that our method is arguably close to the maximum that can be obtained. Note that there may be noise in the dataset, incorrect annotations, that are impossible to find with a correct entity linking system. This would bring the maximum achievable scores even lower.

Table 6.4: Entity Disambiguation Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$ score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Graph Walk</td>
<td>82.45%</td>
<td>74.17%</td>
<td>78.31%</td>
</tr>
<tr>
<td>$e = \arg\max_e p(e</td>
<td>n)$</td>
<td>83.74%</td>
<td>74.14%</td>
</tr>
<tr>
<td>Loopy Belief Propagation Calibrated</td>
<td>88.54%</td>
<td>78.45%</td>
<td>83.49%</td>
</tr>
<tr>
<td>Maximum Achievable</td>
<td>100%</td>
<td>89.38%</td>
<td>94.69%</td>
</tr>
</tbody>
</table>

6.3.3 Overall Entity Linking Performance

Putting both phases together and by varying the keyphraseness threshold, we control the precision–recall trade-off of our entity linking system. We compare the best performing variant of loopy belief propagation – calibrated and scaled, without the $\beta$ parameter, with the random graph walk baseline and the most frequent entity baseline. The results can be seen in the
6. Experiments and results

precision–recall graph in Figure 6.3.

Figure 6.3: Overall results of our method and our baselines.

Note that, for previously stated reasons, the reported results are obtained by using the pruned anchor text index.

As can be seen in the figure, all methods have competitive performance. The mention detection performance puts an upper bound on the obtained results and there is a small gap between the maximum and the obtained results of each method. Occasionally, at certain keyphraseness levels, our method outperforms the other methods on the IITB dataset.
Conclusion and Future work

In this thesis we have investigated a novel approach for solving the entity linking task: applying the loopy belief propagation inference method on a graphical model that captures our assumptions regarding the structure of the problem.

We have demonstrated how potential functions can be calibrated on observed data and then integrated in a joint probability model. We have made incremental improvements to our model and we have carried out experiments that prove their efficiency.

We have empirically shown that our method outperforms both baselines on the entity disambiguation task, on our chosen dataset and that our method performs arguably close to the maximum achievable scores. On the whole task, we have empirically proven that our method has competitive performance, while comparing with both baselines.

Nevertheless, there is still room for improvement, in particular, in the mention detection phase. Future work might investigate starting from a core set of highly confident mentions, for instance the ones with high keyphraseness scores, disambiguating them, and then adding more mentions that are related to the identified entities. The process can then be repeated in rounds, by lowering the keyphraseness scores to introduce new mentions, filtering out the mentions that are not connected to the current set of detected entities and applying loopy belief propagation to find the most likely assignment of entities for the new mention set.

Moreover, a larger corpus can be used for testing, like the massive ClueWeb12 dataset. The Freebase annotations that are available for the corpus can be used to compare against the results obtained with our method. Disagreements can be sampled and manually evaluated in order to acknowledge which method is better in which percentage of the time.
7. Conclusion and Future work

Further, weights can be integrated in our model with the purpose of attributing higher contributions to important mentions. For instance, the random graph walk method assigns importance scores to each of the mentions based on their tf-idf scores. We can integrate these weights in our joint probability model and evaluate the obtained performance.

We hope that this work will lead to improvements on future entity linking solutions and we look forward to further develop this system.
Bibliography


[23] Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In ACL, pages 1375–1384, 2011.


Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor’s thesis, Master’s thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

LOOPY BELIEF PROPAGATION FOR LARGE-SCALE COLLECTIVE ENTITY LINKING

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):
HORLESCU

First name(s):
MARINA

With my signature I confirm that
− I have committed none of the forms of plagiarism described in the ‘Citation etiquette’ information sheet.
− I have documented all methods, data and processes truthfully.
− I have not manipulated any data.
− I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date
ZÜRICH, 15 JULY 2014

Signature(s)