5

Word-Vectors and Search Engines

So far in this book we have discussed symmetric and antisymmetric relationships between particular words in a graph or a hierarchy, described one way to learn symmetric relationships from text, and shown how to use ideas such as similarity measures and transitivity to find nearest neighbours of a particular word. But ideally we should be able to measure the similarity or distance between *any* pair of words or concepts. To some extent, this is possible in graphs and taxonomies by finding the lengths of paths between concepts, but there are at least three problems with this. First of all, finding shortest paths is often computationally expensive and may take a long time. Secondly, we may not have a reliable taxonomy, and as we've seen already, the fact that there is a short path between two words in a graph doesn't necessarily mean that they're very similar, because the links in this short path may have arisen from very different contexts. Thirdly, the meanings of words we encounter in documents and corpora may be very different from those given by a general taxonomy such as WordNet — for example, WordNet 2.0 only gives the *fruit* and *tree* meanings for the word *apple*, which is a decided contrast with the top 10 pages currently returned by the Google search engine when doing an internet search with the query *apple*, which are all about Apple Computers.

Another limitation of our methods so far is that we have focussed our attention purely on individual concepts, mainly single words. Ideally, we should be able to find the similarity between two *collections* of words, and quickly. For this, we need some process for *semantic composition* — working out how to represent the meanings of a sentence or document based on the meaning of the words it contains. This all sounds like a

pretty tall order, but (to some extent) it's actually been possible for years and you will probably have encountered such a system many times — it's precisely what a search engine does.

The traditional ways in which a search engine accomplishes these tasks are almost entirely mathematical rather than linguistic, and are based upon numbers rather than meanings. Using its distribution in a corpus, the meaning of each word is represented by a characteristic list of numbers, and the numbers representing a whole document are then given simply by averaging the numbers for the words in the document. Bizarre but effective!

This chapter is all about the lists of numbers used to represent words and documents, and these lists are called *vectors*. The discussion of vectors will finally bring us to talk in detail about different *dimensions* and what they mean to a mathematician. This in itself might be of interest to many readers, and quite different from what you supposed.

5.1 What are Vectors

A *vector* is a very useful way of keeping track of several different pieces of information, all of which relate to the same concept or object. For example, suppose I have a drawer at home where I keep my leftover currency from travelling to different countries, and have a small pile of 6 US dollars ($6), 20 UK pounds (£20), 15 Euros from Germany (€15) and 100 Japanese yen (¥100). We could write this information down in a table

$	£	€	¥
6	20	15	100

or if we keep careful track of the convention ($, £, €, ¥) we could write this as the *row vector*

$$Dm = (6, 20, 15, 100),$$

where Dm stands for "Dominic's money." Now, suppose that Maryl has a similar drawer, which contains

$	£	€	¥
50	16	20	0

This information can be encoded in the row vector

$$Mm = (50, 16, 20, 0).$$

If we marry our fortunes together, what will our combined currency drawer contain? The answer is simple — just add together the number

n the matching positions and you get the combined vector

$$Dm + Mm = (56, 36, 35, 100).$$

Suppose that we decided to put this money into savings and it grew by 20% (which corresponds to multiplying by the number 1.2). Then we'd have

$$1.2(Dm + Mm) = (67.2, 43.2, 42, 120).$$

Not impressed? Maybe it doesn't seem like rocket science to write down two lists of numbers, keep track of which numbers refer to which currencies, and then add and multiply these numbers. But it turns out that the ability to break a situation down into individual numbers, to do separate calculations with those numbers, and then to combine the answers to represent a new situation, can be extremely powerful, because it allows us to break down a potentially complicated process into a number of extremely simple ones. And by the way, you just dealt with points in a *four-dimensional* space without even blinking.

5.2 Journeys in the plane

Another situation that can be described using vectors is one we've already encountered — namely, the 2-dimensional plane can be thought of as a vector space. In Figure 5.1, the arrows a and b represent two 'journeys' in a plane. The combined journey from going along the a arrow and then the b arrow is called the *sum* of the journeys a and b, and so is (naturally enough) written as $a + b$.

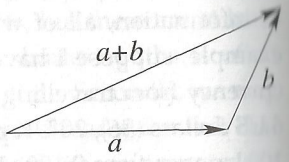


FIGURE 5.1 The journey $a + b$

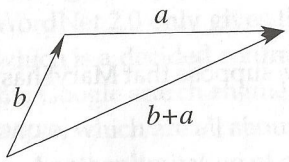
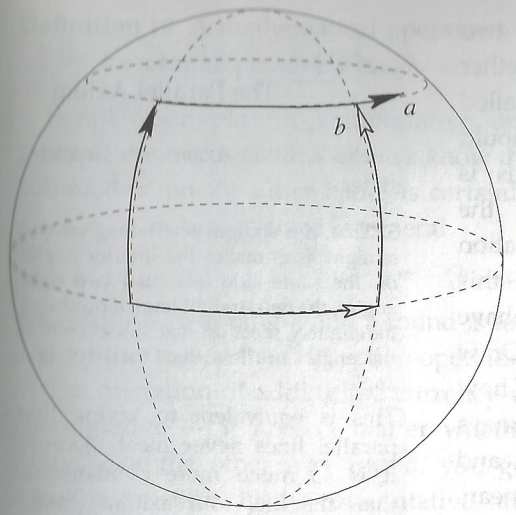


FIGURE 5.2 The journey $b + a$

On the other hand, we could have gone along the journey b first, followed by the journey a, and this combined journey would be called $b + a$. One fundamental property of the flat plane is that these two journeys have the same destination: $a + b$ and $b + a$ are two ways of writing the same journey. This isn't always true — it depends on the space in which a and b are journeys. For example, imagine your space is a sphere like the earth and you start on the equator. If you go 1000 miles north and then 1000 miles east, you will end up further east than if you go 1000 miles east, *then* 1000 miles north, because the parallel (line of latitude) 1000 miles north is a smaller circle than the parallel of the equator, as shown in Figure 5.3.[1] Because a sphere is curved, it turns out that the order in which you make different journeys matters — the convenient identity $a + b = b + a$ ceases to be true.



If you travel north, then east you'll reach point a.

If you travel the same distances east, then north you'll reach point b.

FIGURE 5.3 When combining two journeys on a sphere, you can end up at different places depending on which journey you make first.

In fact, the amount to which different journeys can land you in different destinations can be used to define and measure the very concept of the *curvature* of a mathematical space. Vector spaces are special precisely because they *aren't* curved — they are flat or 'linear' (mathematicians often say *Euclidean*), and because of this the study of vectors is called *linear algebra*. As a result of this linearity, vector spaces obey Euclid's fifth axiom of geometry, which states that parallel lines never meet.

[1] At the extreme, if you go as far as the north pole, the line of latitude collapses from a circle to a single point — the north and south poles do not have a well defined longitude, since *all* lines of longitude intersect at the poles. This singularity provides the solution to the brainteaser

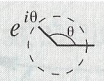
Suppose you walk ten miles south, ten miles east, ten miles north and find yourself at the point you started from. Where are you?

the standard solution being that you are at the north pole. Other solutions lie just north of the south pole.

The process of putting two ɔurneys, or vectors, together ıto a single journey is called *ector addition*. You should onvince yourself that this is wise generalization of the eal-number addition operation ou learnt as a child. In this ontext, real numbers behave s 1-dimensional vectors. Draw ourself a mental picture of how ur $a + b$ and $b + a$ journeys ppear along a single line, and hope you'll see what I mean. ınother way of describing ector addition is called the ɔarallelogram rule. If you can ee why by mentally combining igures 5.1 and 5.2, you're well ın the way to understanding vhat's going on.

The symbol + which means add together these two vectors or numbers)" is called an *perator*. An operator is different rom a relationship such as imilarity $\leftrightarrow$ or hyponymy $\sqsubseteq$, ɔecause whereas $a \leftrightarrow b$ is just a tatement that the relationship ıolds, the operation $a + b$ has a result or an outcome. Familiar examples ɔf relationships between numbers are equals ($=$) and less than ($<$). ?amiliar examples of operators are multiplication ($\times$), addition ($+$) and ubtraction ($-$). The similarity and distance measures of Chapter 4 are ıll operators because they produce a number as their outcome.

Just as a relationship $a \sim b$ where the order of a and b is interchangeable is called symmetric (Definition 2), an *operator* for which the ɔrder of the inputs is interchangeable also has a special name.



The Parallel Axiom

Euclid's fifth *Axiom of Geometry* states

5. That, if a straight line falling on two straight lines makes the interior angles on the same side less than two right angles, the two straight lines, if produced indefinitely, meet on that side on which the angles are less than the two right angles.

This is equivalent to saying that parallel lines never meet. Because it is so much more cumbersome than the first four axioms, mathematicians tried to prove it as a consequence of these, rather than assuming it as an axiom in its own right, for over two thousand years. In the 19th century, mathematicians such as Gauss, Bolyai, Labachevsky and Riemann finally realized that the behaviour of parallel lines depends on the nature of the space you are working in: for example, all the lines of longitude on a globe are parallel at the equator and meet at the poles. Letting go of the parallel axiom led to a whole new field of *non-Euclidean geometry*, which paved the way for the Theory of Relativity.

Definition 10 A mathematical operator $\circ$ is said to be *commutative*[2] if $a \circ b = b \circ a$ for all possible a and b.

Simple examples of commutative operations are addition and multiplication of real numbers: we know that $a + b = b + a$ and $ab = ba$. Subtraction, on the other hand, is certainly *not* commutative, because $a - b \neq b - a$. In fact, since in general

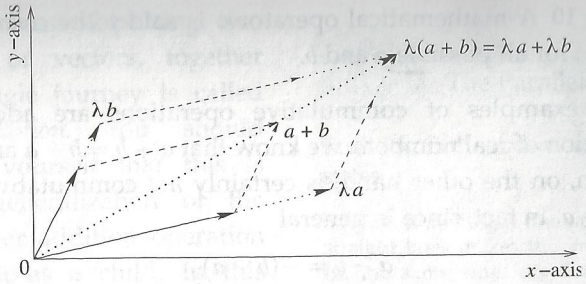
$$a - b = -(b - a),$$

the result of swapping a and b round is to give us the *opposite* answer, and for this reason the subtraction operator is *anticommutative*.

The operation of adding the currency vectors in Section 5.1 is also commutative — it doesn't matter whether you add my fortune to Maryl's, or the other way round, you get the same answer. Vector addition must always be commutative, and this is one of the reasons why vector addition is a good generalization of real number addition. In fact, the operation of amalgamating of our currency vectors is commutative precisely because it is a combination of four separate addition operations with numbers, and each of these separate sums is commutative.

So much for adding vectors. The other operation we must be able to perform is *scalar multiplication* — stretching or shrinking of vectors. Any of our journey vectors can be scaled up or down to give you a journey in the same direction but of different length. Similarly, we could scale our currency vector by a factor of 20% (multiplying by 1.2), or any other factor (though in this case we might find ourselves rounding the result to two decimal places or the nearest cent, which the bank normally does on our behalf when calculating interest). Just as addition must satisfy a few properties such as being commutative, scalar multiplication must also be well-behaved in certain ways. For example, if Maryl and I invested our money separately, each got a return of 20%, and then married our new fortunes together, we should get exactly the same amount as if

[2]The word *commutative* has nothing to do with travelling to work or having a prison sentence reduced: nor is there really any good reason why we couldn't call an operator *symmetric* instead of coining this new technical term. There is no particularly good reason why you have to learn a new four syllable word at this point, except that for those readers who are already familiar with it, it would be too confusing to change things now. (Because of this, the excessive jargonization of all contemporary fields is getting out of hand and is apparently impossible to resist, the proliferation of science itself scattering us in a technological Tower of Babel.)



GURE 5.4 Adding two vectors together and then rescaling gets you to the same place as rescaling both vectors first and adding the results. This is another characteristic of Euclidean (flat) space.

re married them together first and then increased the total by 20%. In quations, this would be

$$1.2Dm + 1.2Mm = 1.2(Dm + Mm),$$

which you can easily check is true in this case. An illustration of this roperty is given in Figure 5.4. This scaling operation can be traced back o Euclid's second Axiom of Geometry, which states that it is always ossible

2. To extend a finite straight line continuously in a straight line.

A set of points equipped with addition and scalar multiplication perations must satisfy the eight axioms of Figure 5.5 to be a *vector space*. hese axioms were given by Peano in 1888, though this was really a ulmination of different strands of mathematical progress throughout he 19th century and before. In the next section we will explain how his culmination came to be, and this process will hopefully make some difficult ideas very clear. This will leave us in a sound position to use the echniques of vector spaces to describe words and their meanings.

5.3 Coordinates, bases and dimensions

What do the currency vectors of Section 5.1 and the journey vectors of Section 5.2 have in common? We could try to go through and check that the eight axioms of Figure 5.5 hold for both systems and say that this *is* what they have in common, but this is really a topic for a linear algebra homework assignment, not a book on meaning. What we want to know is how they came to be regarded as similar systems — after

Formal Definition of a Vector Space

A (real) Vector Space is a set V equipped with two mappings, called addition (which maps $V \times V$ to V) and scalar multiplication (which maps $\mathbb{R} \times V$ to V).

Addition must obey the following axioms:
- A1. Addition is associative, so that for all $a, b, c \in V$, $(a + b) + c = a + (b + c)$.
- A2. Addition is commutative, so that for all $a, b \in V$, $a + b = b + a$.
- A3. There is an additive identity element $0 \in V$ (called the "zero vector") such that for all $a \in V$, $a + 0 = 0 + a = a$.
- A4. For each element $a \in V$, there exists an element $-a \in V$ with $a + (-a) = 0$.

Scalar Multiplication must obey the following axioms:
- M1. Scalar multiplication is associative, so that for all $\lambda, \mu \in \mathbb{R}$ and for all $a \in V$, $(\lambda\mu)a = \lambda(\mu a)$.
- M2. $1a = a$ for all $a \in V$.
- M3. Scalar multiplication is distributive over addition in V, so that for all $\lambda \in \mathbb{R}$ and for all $a, b \in V$, $\lambda(a + b) = \lambda a + \lambda b$.
- M4. Scalar multiplication is distributive over addition in $\mathbb{R}$, so that for all $\lambda, \mu \in \mathbb{R}$ and for all $a \in V$, $(\lambda + \mu)a = \lambda a + \mu a$.

FIGURE 5.5 A vector space must satisfy these eight axioms (Jänich, 1994, p. 17)

all, this similarity was recognized long before the axioms were written down.

The main breakthrough in thinking of journeys in the plane as lists of numbers is made by giving each journey a set of *coordinates*, which break the journey down into different components in different directions. You will almost certainly have come across these in high school, and we have used them already in this book.

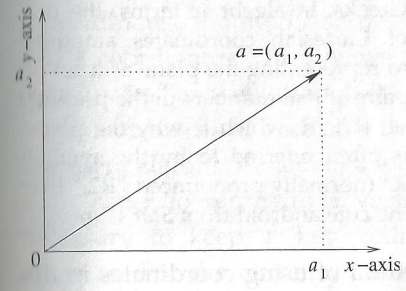


FIGURE 5.6 The coordinates of a vector

The basic picture is in Figure 5.6, where the vector a is decomposed into 2 parts by *projecting* it onto two fixed vectors or *axes*. (The word *project* comes to us through the Medieval Latin *projectum*, meaning *throw forward*.) If the projection hits the x-axis at a distance of a_1 units from the origin, this number is said to be the *x-coordinate* of the point a, and similarly, if the projection hits the y-axis at a_2, then this is said to be the *y-coordinate* of a. The point a is then said to have coordinates (a_1, a_2),

hough it is only by convention that the first coordinate refers to the orizontal component and the second to the vertical component, the right' and 'up' directions regarded as positive.[3]

There is one confusion to void. We've talked about giving oordinates to points in the plane nd to journeys in the plane. ut a journey requires a pair of oints (a beginning and an end), ot a single point, so how can oth these ideas be represented y the same sort of coordinates? he answer is that the point a is dentified with the journey *from he origin* or zero point to the oint a, so we can think of both 'the point a" and "the journey rom 0 to a" as vectors in the lane.

Giving these numbers or coordinates to a journey vector nables us to carry out the rocesses of vector addition and calar multiplication without aving to draw the pictures. or example, we can work out he additions in Figures 5.1 and 5.2 very easily: if the vector a as coordinates (a_1, a_2) and the ector b has coordinates (b_1, b_2) hen their sum $a+b$ will have the coordinates $(a_1 + b_1, a_2 + b_2)$. The method of using coordinates in this ashion was a gradual mathematical development: Appolonius of Perga ca.260-190 BC) and the French bishop Nicole Oresme (ca.1320-1382)

Cartesian coordinates

The use of a pair of numbers such as (x, y) to represent points in the plane is justly associated with Rene Descartes (1596-1650), though he was not the first to invent it. Descartes realized that many of the relationships between geometric figures such as lines and curves could be represented by numbers in this way — for example, the cosine wave on page 106 is the locus of all points whose Cartesian coordinates (x, y) are related by the equation $y = \cos(x)$.

In this way, many old geometric problems, such as finding the point where two curves intersected, could be solved using the techniques of algebra which, thanks mainly to Islamic mathematicians, had made enormous progress since the Greeks. In algebraic terms, the use of Cartesian coordinates amounts to representing the plane as a set of pairs of real numbers in the product set $\mathbb{R} \times \mathbb{R}$, which is why the plane is often referred to by the symbol $\mathbb{R}^2$ (normally pronounced "R2," like the cute android from *Star Wars*).

[3]That this is only conventional is apparent in computer graphics, where the 'down' direction is normally regarded as positive and the position of a point is determined by its coordinates measured from the top left hand corner of the screen.

both described points in figures using distances from a pair of fixed locations, ideas which contributed to the *Analytic Geometry* of Descartes (1637). The description given by Descartes of choosing a particular line as a unit, and ascribing numerical lengths to other lines according to their ratio with this unit, is one of the first clear-cut definitions of 'measurement' in the sense of Section 1.4.3.

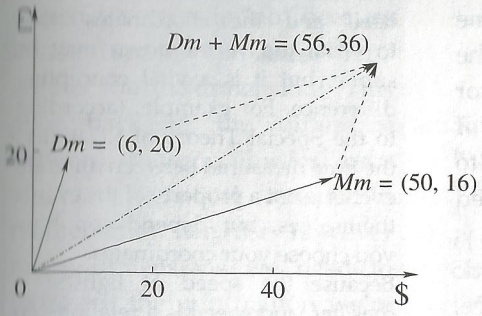


FIGURE 5.7 Adding together currency vectors

This explains how we can use numbers to represent the pictures of Figures 5.1 and 5.2. We can also go the other way — suppose we wanted to represent the first two coordinates of the currency vectors in Section 5.1 pictorially. (These are the amounts of dollars and pounds, $Dm = (6, 20)$ and $Mm = (50, 16)$.) We could plot these on a grid and add them together using the parallelogram rule, and finally read off the coordinates of the resulting sum, $Dm + Mm = (56, 36)$, as in Figure 5.7. This may seem like a long way round, though as we'll see in Chapter 6, representing vectors as points on a plane like this, rather than as a pair of numbers, can give a much more intuitive representation of which groups of vectors are close to one another. Just as points in the plane are often given x and y coordinates, Figure 5.7 shows that we can just as well talk about "$ and £ coordinates." Ideally, we would have represented the "€ and ¥ coordinates" as well, though it would be difficult to represent three of these coordinates at once on a flat page, and pretty much impossible to represent all four of them in a way that made intuitive sense.

In order to represent a vector as a list of numbers, it is clearly necessary to keep a 'key' telling us which quantity each number refers to. For the currency vectors, we needed to remember that the code (a_1, a_2, a_3, a_4) was shorthand for "a_1, £a_2, €a_3 and ¥a_4." When working in the plane we need to choose coordinate axes, and while the convention is to use 'eastward' and 'northward' pointing axes, there is no *a priori* reason for making this particular choice.[4] This key telling

[4]We could use 'south' and 'north-northwest' as our coordinate axes, since you can

us which coordinate is which is called a *basis* for the vector space. For example, the basis for our currency vectors is the set $\{\$1, £1, €1, ¥1\}$.

Now, each coordinate refers back to one of these basis elements, so the number of coordinates needed to represent any point is always the same as the number of items in the basis. For example, the vector $Dm = (6, 20, 15, 100)$ has 4 coordinates, one to refer to each of the 4 currencies listed in the basis. This number is an important characteristic property of any vector space, and it's a word you will have come across many times.

Definition 11 The *dimension* of a vector space is the number of coordinates needed to specify a given point uniquely.

The flat plane has two dimensions, since each point needs 2 coordinates to represent it: normally these are the x and y coordinates. The currency vectors of Section 5.1 have *four* dimensions, because each vector is made up of 4 coordinates (or numbers) of Dollars, Pounds, Euros and Yen.

Time: absolute or relative?

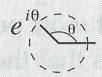
The distinction between a vector itself and the coordinates used to write the vector down may be subtle, but it is a vital conceptual difference. For example, (according to the Special Theory of Relativity) the time measured between different events is not a property of the events themselves, but depends on how you choose your coordinates.
Because the speed of light c is *constant*, and speed is a relationship between position and time, if you change the way you measure position you must also change the way you measure time. The relationships between the time coordinates of different events is therefore different for different observers. This interplay between space and time has strange consequences — for example, two events which are simultaneous for one observer may happen at *different* times for another. If this sounds confusing but you really want to get it, see `www.puttypeg.com/train.html`.

still specify any journey by saying how far south you have to go, followed by how far north-northwest you have to go — and this representation would be just as unique and unambiguous as the more familiar x and y coordinates. When developing the technique of using coordinates, Descartes in fact used many different pairs of axes, not necessarily at right angles to one another — the convention of using the same pair of perpendicular axes for most problems came later.

Hopefully this will break a few misunderstandings, if they haven't been broken already. We've probably all heard and wondered about the question

Is time the fourth dimension?

Clearly the answer for our currency vectors is "no" because in this space, "Japanese Yen" is the fourth dimension! This goes back to the difference between studying physical *space* and mathematical *spaces*. In mathematics, we use however many dimensions we need to represent the information we're working with — if a situation is complicated enough to require more than than 2, or 3, or 4 coordinates to represent it properly, then we'll use a space with as many coordinates as are necessary. However, if you *are* seeking to model the physical universe, then four coordinates can be a very good choice — three to measure different directions in space, and one to measure the time that elapses between two events (this is exactly what we did when defining the *Minkowski distance* on page 110). So in this sense, the answer is "yes," insofar as there are some very good models of the physical universe in which time is a fourth dimension.



Coordinates and vectors

Since there are many different ways to choose a basis, and thus to assign coordinates to each point in a vector space, we must check that the mathematics is the same whatever choice we make. For example, if the dimension of a vector space is the number of elements in any basis, we had better make sure that this number is the same for *every* possible basis of a given vector space (Jänich, 1994, p. 46) — otherwise 'dimension' would not be a property of the space itself, but only of one particular coordinate system.
Also, the sum of two vectors $a + b$ can be calculated by adding their respective coordinates, but must be the same vector for every possible coordinate system. These important properties can all be proved from the axioms in Figure 5.5.

However, just because a fourth dimension is a useful concept, it doesn't make it an *easy* concept, and the difficulties have been noted since vectors were first invented. The term *vector* was probably first used by Sir William Rowan Hamilton in his work on vectors in three and four dimensions (which he called *quaternions*). His verbal description

of how a fourth direction might be perpendicular to the North-South, East-West *and* Up-Down directions all at the same time is quite detailed and at the same time almost desperate (Hamilton, 1847). The situation is touchingly parodied in the novel *Flatland* (Abbott, 1884, Ch. 21), where a square vainly tries to coax his hexagonal grandson into the speculation that there might be a direction that is "Upward and not Northward."

The main trouble that most people have when trying to handle more than three dimensions is that they try to visualize all these dimensions at once. We already accepted this limitation in Figure 5.7: when I was explaining what currency vectors and journey vectors have in common, I only drew a picture of two of the currency coordinates (dollars and pounds) since we could easily represent these on a flat piece of paper. Do not hold yourself back from understanding what dimensions are by trying to visualize more than three dimensions at once: it's a good way of getting frustrated because it's just not within our physical experience. Spaces with many dimensions are easy to live and work with once you accept that several dimensions are perfectly valid and consistent, without trying to forcibly reconcile each dimension with visual experience simultaneously.

5.4 Search Engines and Vectors

Early search engines relied on simple Boolean matching algorithms, which will be described in Section 7.1. If a document matched the keywords in a user's query, it was marked relevant and returned to the user: if not, it was marked irrelevant and left behind. But by the 1970's, as document collections grew bigger, the flaws in such an approach became more and more damaging. One main problem that people encountered is that with large document collections (which would be considered small by today's standards), returning *every* document that matched the keywords would still leave the user with a huge mass of documents to wade through to try and work out if they were relevant. For example, suppose that you wanted to find out about helicopters. A search of the internet that returned *every* matching document might send you countless news articles describing rescue missions that involved a helicopter at some point, but these may not tell the you very much about the helicopters themselves (though they may tell you something about the contexts in which helicopters appear).

TABLE 5.1 Recording the number of times each indexed term occurs in each document

	Document 1	Document 2	Document 3
bank	0	0	4
bass	2	4	0
commercial	0	2	2
cream	2	0	0
guitar	1	0	0
fishermen	0	3	0
money	0	1	2

To take more extreme examples, a dictionary will probably contain most common words that aren't proper names, but the dictionary isn't relevant to every user whose query contains any of these words. A telephone directory will contain a huge number of proper names, but it still won't be relevant to many queries containing one of these names. Long documents containing huge numbers of words could be declared relevant to almost any query — but these are often precisely the documents whose information is most diluted. It became very clear that a better measure of relevance was needed than could be achieved by simply partitioning the document collection into documents that did and didn't contain the keywords.

One way to do this (and I should stress that there are several) is to represent words using vectors. The coordinates of these vectors are numbers which measure the extent to which a particular word is important to a particular document. To begin with, these coordinates are obtained simply by counting the number of times each word is used in each document.

As a small-scale example of this technique, consider the three documents in Figure 5.8, in which a handful of different words have been highlighted (*bank, bass, commercial, cream, guitar, fishermen* and *money*). The number of instances (tokens) of each of these words in each of the three documents is recorded in Table 5.1.

Such a table, which records the number of times each word occurred in each of a collection of documents, is called a *term-document matrix*.[5]

[5] A *matrix* (plural *matrices*) in this context is simply a table of numbers. There are well-defined algebraic rules for how pairs of matrices can be added and multiplied together, provided that the number of rows and columns in the two matrices is compatible.

Document 1 (BNC)
The first **Bass** VI I remember seeing was being used on television by Jack Bruce, during a mimed performance of Strange Brew by **Cream**. Recollections indeed: an extremely fashionable **Cream**, with serious sideburns all round, grossly extended collars on satin shirts, Clapton's Hendrix perm, flared trousers and Les Paul. Being nobbut a sprog, I remember wondering why bassist Bruce was playing a **guitar**. Only he wasn't, of course; he was playing a Fender **Bass** VI. He also became arguably the most famous exponent of the instrument, along with Eric Haydock of The Hollies.

Document 2 (NYT, 1996)
ORLEANS, Mass. — When weekend **fishermen** looking to unwind come to Rock Harbor, they take out their rods and reels and angle for striped **bass** one by one. When **commercial fishermen** like Mike Abdow go out on their boats to earn a living, they catch **bass** the same way. By law, they must reel them in one at time, without nets, traps or long-line gear. But right now, the waters are rough between people who fish **bass** for a living and those who angle for pleasure. In a feud that has divided people along the Massachusetts coast, big-**money** sporting interests are trying to stop small-time **commercial fishermen** from pulling in any more striped **bass**.

Document 3 (NYT, 1995)
Kuala Lumpur, Oct. 26 (Bloomberg) – **bank** Negara will change the way it calculates the cost of lending **money** to **commercial** banks and financial institutions. In a statement, the **bank** said that as of Nov. 1, the base lending rate, at which **commercial** banks can borrow from the central **bank**, will become more responsive to movements in **money**-market rates. Several weeks ago, the **bank** said it would change the way it calculates the base rate, said Desmond Ch'ng, a banking analyst OCBC Securities (Malacca) Sdn. Bhd.

FIGURE 5.8 Three documents about different topics

The snapshot we've taken in Table 5.1 is only a tiny fragment of the term-document matrix used by any real search engine — we've only taken three documents and only considered a few of the words they contain. However, it's still possible to get some idea of how the table works, and hopefully the reader will then be able to extrapolate and imagine the huge table we'd produce from a document collection of several million words or even several billion webpages.

Table 5.1 can be used as a simple inverted file index, like a traditional concordance.[6] That is, if you're interested in *fishermen*, the table will tell you to look in Document 2, if you're interested in a *bank*, the table will tell you to look in Document 3, and so on. But the real advantage of measuring the *number* of times each word occurs in a document comes when you have words occurring in many documents, and you want to know which document is the most relevant. For example, if you want to find out about *money*, Table 5.1 will tell you that it's mentioned in both Document 2 and Document 3, but since it's mentioned *twice as much* in Document 3, you should start by reading this document, because the term *money* is more concentrated or denser in this document. This would be good advice — while Document 2 mentions tensions between two groups of people which have monetary consequences, Document 3 is directly about finance.

There are many problems and issues with this idea that we haven't even begun to address, some of which are listed below:

- Many possible meanings of the different terms are absent from this fragment — *bank* can also refer to a *river bank*, and most of the time *cream* is more likely to mean a *dairy product* than a *1960's rock band*.
- The term *bass* does have more than one meaning in our 3 documents — it means a kind of *musical instrument* in Document 1 and a kind of *fish* in Document 2. If a user is only interested in one of these meanings (and in this case, it's unlikely that they'd be interested in

Matrices grew out of an an 1858 memoir on transformations written by the British mathematician Arthur Cayley (1821-1895), partly as a generalization of Hamilton's *quaternions* (Boyer and Merzbach, 1991, Ch. 26).

[6]The term 'inverted file index' or 'inverted file format' is used because words are described by a list of documents, whereas it's more usual to think of documents being described by a list of words (Salton and McGill, 1983, Ch. 2). A concordance, often of the Bible, is a big book where a Biblical scholar can look up a word such as *light* and find the chapter and verse of every place where the word *light* appears, hoping to trace the way a theological concept is used and developed through the centuries (Witten et al., 1999, Ch. 1).

both at the same time), how are we to enable users to search for only the documents containing this meaning of *bass*?

- The term *guitar* only appears once in Document 1 — but it's very possible that a user searching for the term *guitar* would also find articles containing the term *Les Paul* relevant, since *Les Paul* is a make of *guitar*.

Many of these questions will be addressed during the rest of this book — though in truth, none of them can honestly be said to have been completely solved.

Now, here's the point. The rows of numbers in Table 5.1 can be thought of as vectors, just like the lists of numbers in the 'currency vectors' of Section 5.1. The different rows can be added together component by component, or scaled by any other real number, and it is a simple matter to check that these rows, and the operations of row addition and scalar multiplication, satisfy the axioms of Definition 5.5. Because of this, some of the theory and techniques of vectors, which are very well-developed and well-understood parts of mathematics, can be used to calculate and reason with words.

In fact, because they only have 3 coordinates each, it's almost possible to imagine these particular word vectors as points in a 3 dimensional space, as I've tried to depict in Figure 5.9. However, as we go through this section you'll realize that this diagram is only a visual aid — all of the mathematics we use to work out which words are similar to which other words can be done entirely by working with the coordinates in Table 5.1.

5.5 Similarity and distance functions for vectors

In this section we describe the most prominent ways to measure similarity and distance between vectors with any number of dimensions, and show how they can be applied to our word vectors. This should provide you with some robust mathematical tools and terminology (which may seem challenging, but which are easy to use in practice and very easy to program into a computer) and a taste of the possible linguistic applications.

We start with a few examples from the term-document matrix in Table 5.1, which gives word vectors such as

$$\textit{bank} = (0, 0, 4), \quad \textit{bass} = (2, 4, 0) \quad \text{and} \quad \textit{money} = (0, 1, 2).$$

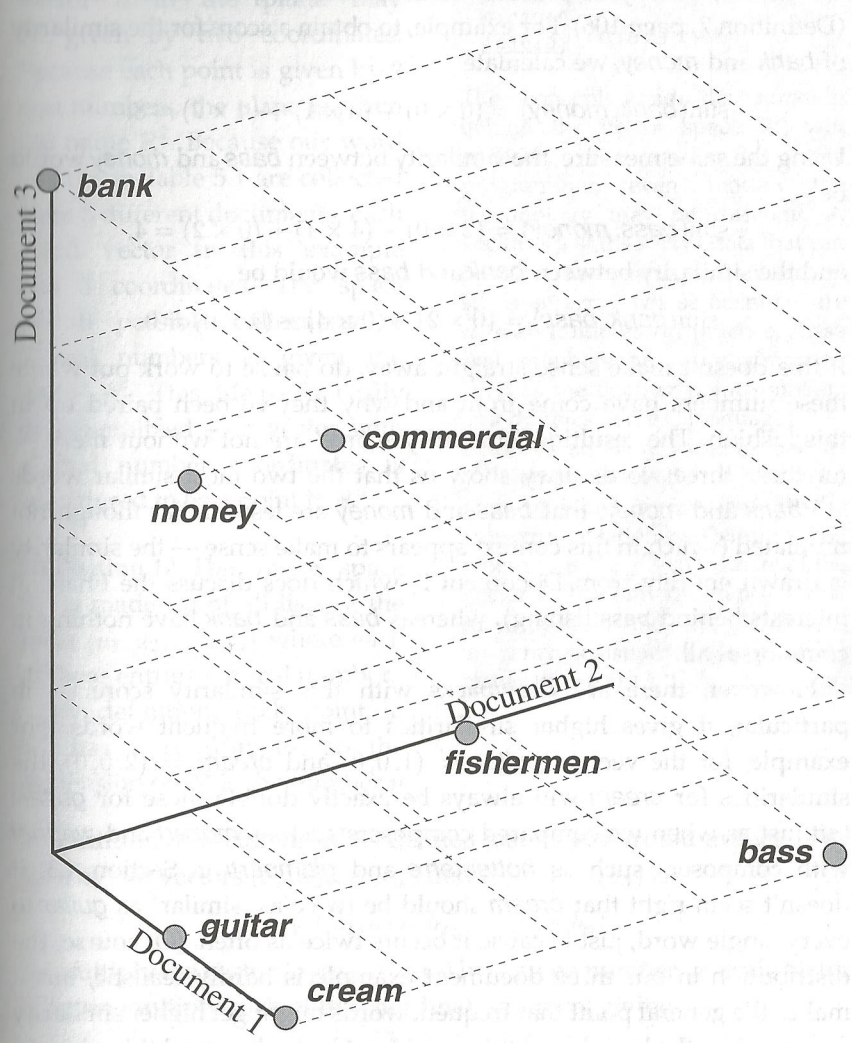


FIGURE 5.9 Imagining the word vectors of Table 5.1 as points in the three-dimensional space $\mathbb{R}^3$

In order to measure similarity between these word vectors, we could just multiply their first, second and third coordinates together and add the results, exactly as we did when introducing the cosine measure (Definition 7, page 106). For example, to obtain a score for the similarity of *bank* and *money*, we calculate

$$\text{sim}(\textit{bank}, \textit{money}) = (0 \times 0) + (0 \times 1) + (4 \times 2) = 8.$$

Using the same measure, the similarity between *bass* and *money* would be

$$\text{sim}(\textit{bass}, \textit{money}) = (2 \times 0) + (4 \times 1) + (0 \times 2) = 4,$$

and the similarity between *bank* and *bass* would be

$$\text{sim}(\textit{bank}, \textit{bass}) = (0 \times 2) + (0 \times 4) + (4 \times 0) = 0.$$

If this doesn't make sense straight away, do pause to work out where these numbers have come from and why they've been paired up in this fashion. The resulting 'similarity scores' are not without merit — for these three words, they show us that the two most similar words are *bank* and *money*, that *bass* and *money* are less similar though not unrelated (which in this context appears to make sense — the similarity is drawn entirely from Document 2, which does discuss the financial interests behind bass fishing), whereas *bass* and *bank* have nothing in common at all.

However, there are drawbacks with this similarity score — in particular, it gives higher similarities to more frequent words. For example, for the vectors *guitar* $= (1, 0, 0)$ and *cream* $= (2, 0, 0)$, the similarities for *cream* will always be exactly double those for *guitar*: but just as when we compared composers such as *mozart* and *wagner* with composers such as *hottenterre* and *giamberti* in Section 4.3, it doesn't seem right that *cream* should be twice as 'similar' as *guitar* to every single word, just because it occurs twice as often. (Of course, the distribution in our three document example is hardly realistic, but it makes the general point that frequent words might get higher similarity scores across the board, and this would not be such a good thing.)

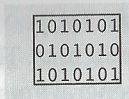
Just as in Section 4.3, we can get round this problem by *normalizing* or dividing out our similarity scores. This will be easiest if we introduce some notation and terminology for vectors in general, which will also be frequently used in later chapters.

5.5.1 Introducing the vector space $\mathbb{R}^n$

So far we've seen that a journey vector a in the plane may be given by two coordinates. Because each point is given by 2 real numbers, the plane is given the name $\mathbb{R}^2$. Because our word vectors in Table 5.1 are collected from 3 different documents, each word vector in this example has 3 coordinates. The space of all possible collections of 3 real numbers is given the name $\mathbb{R}^3$. This idea can easily be generalized — a vector with n real number coordinates is considered to be a point in $\mathbb{R}^n$.

Definition 12 The vector space $\mathbb{R}^n$ is made up of all lists of the form $(a_1, a_2, \ldots, a_n)$ where each of these entries is a real number.

By definition, each point in $\mathbb{R}^n$ has n coordinates, so the dimension of $\mathbb{R}^n$ is the number n.



Arrays

The concepts and the notation behind the vector space $\mathbb{R}^n$ will probably be familiar to any programmer, even though the terminology may be different. A vector is just the sort of data that can be stored as an *array*, and a vector in $\mathbb{R}^n$ is an array whose elements are n 'real' numbers (in practice, these real numbers are approximations such as the floating point numbers and doubles of the C language).

If `arr` is an array variable then its j^{th} element is usually referred to by `arr[j]`, which is very similar to the subscript notation of Definition 12, where a_j is the j^{th} coordinate of the vector a. The only difference is that in computing, the first element of an array is usually called `arr[0]` rather than `arr[1]`.

Addition of vectors in $\mathbb{R}^n$ is defined just as you would expect — the sum of two vectors $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ is

$$(a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n).$$

To multiply the vector $(a_1, a_2, \ldots, a_n)$ by a given number or 'scale factor' λ, again multiply each of the coordinates in turn, giving

$$(\lambda a_1, \lambda a_2, \ldots, \lambda a_n).$$

This may look horrible. If you're having difficulty, go back to the calculations with currency vectors in Section 5.1, which are precisely examples of the equations above. Because we're so used to the idea of currencies, it's completely natural to think of adding two collections

together by adding the dollars to the dollars, then adding the pounds to the pounds, etc. It also makes sense that to scale a whole currency vector by a given scale factor, you have to scale each currency (or each coordinate) by this factor in turn. The equations we've just written down are nothing more than general versions of exactly these 'one currency at a time' operations.

Once we start dealing with more than two or three coordinates, we normally stop using different letters such as (x, y, z) for the different coordinates, for two reasons. The first is that sooner or later we'd run out of letters. The second is that it's actually *easier* to use the subscript notation of Definition 12. How it will normally work is that we'll use one letter for each whole vector (such as a), and subscripts such as a_1, a_2, *etc.* for the coordinates. In this way, there are fewer letters to keep track of, and you know straight away that (for example) a_3 means "the 3rd coordinate of the vector a." This sort of notation makes it very easy to extend the Euclidean distance and cosine similarity measure of Chapter 4 to cope with any number of coordinates.

5.5.2 The scalar product of two vectors

Now that we're becoming familiar with vectors in $\mathbb{R}^n$, and the way of writing the coordinates $(a_1, a_2, \ldots, a_n)$ for the vector a, we can use these techniques to derive equations for computing similarities and distances for general vectors. We've already used a form of similarity score between our word vectors, by multiplying their coordinates and adding the results. This 'product' of two vectors has a special name.

Definition 13 Let $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_n)$ be vectors in $\mathbb{R}^n$. Their *scalar product* $a \cdot b$ is given by the formula

$$a \cdot b = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n.$$

The scalar product is sometimes called the *inner product* or just the *dot product* of two vectors. We'll see shortly that the scalar product is closely linked to both the Euclidean distance and cosine similarity measures of Chapter 4.

5.5.3 Euclidean distance on $\mathbb{R}^n$ — extending Pythagoras' theorem to n dimensions

The two-dimensional version of Pythagoras' theorem (Section 101) will already be familiar to most readers because it is widely taught in schools. This enables you to work out the length of the hypotenuse of a right-angled triangle (alternatively, the length of the diagonal of a rectangle).

A less well-known but equally interesting fact is that exactly the same principle can be used in *three* dimensions to calculate the length of the diagonal of a solid box or 'cuboid' (which is the three dimensional version of a rectangle). For example, if the three perpendicular sides of a box have lengths p, q and r, then the diagonal of the box will have a length of $\sqrt{p^2 + q^2 + r^2}$, which is easy to prove.[7] If the corners of the box are the points a and b, which have coordinates (a_1, a_2, a_3) and (b_1, b_2, b_3) respectively, then the lengths of these sides are $p = b_1 - a_1$, $q = b_2 - a_2$ and $r = b_3 - a_3$. The length of the diagonal, which is the distance $d(a, b)$ between a and b, is therefore given by the formula

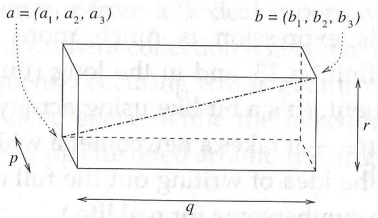


FIGURE 5.10 Measuring the diagonal from a to b.

$$d(a, b) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + (b_3 - a_3)^2}. \qquad (5.10)$$

This is essentially the same as the two-dimensional distance formula given in equation (4.6), though we've changed some of the notation. We can also use the summation sign Σ (the capital Greek letter sigma) to mean "the sum of all these numbers," in which case equation (5.10) can be written using the shorthand

$$d(a, b) = \sqrt{\left(\sum (b_i - a_i)^2\right)}, \qquad (5.11)$$

where the subscript i stands for each of the indices in turn. This means exactly the same thing as equation (5.10), once you've expanded out the terms in the $\sum$ expression. (If we wanted to make it particularly explicit

[7] The proof works by applying Pythagoras' theorem twice: first consider the diagonal of the rectangular base of the cuboid, whose length is $\sqrt{p^2 + q^2}$; and then consider the hypotenuse of the triangle made by this diagonal and the upright distance r.

that the subscript i takes the values 1, 2 and 3 we'd write $\sum_{i=1}^{3}$ though normally this will be unnecessary because it should be clear from the context what range of values the subscript i takes.)

The same index and summation notation we used in equation (5.11) can be used to express the scalar product of two arbitrary vectors $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_n)$ as

$$a \cdot b = \sum a_i b_i.$$

This expression is much more succinct than the first version in Definition 13, and in the long run this brevity of expression is a great benefit. (It's a bit like using acronyms such as USA and UN for special terms — it takes a newcomer a while to get used to, but once you've got it, the idea of writing out the full expression in words every time is far too cumbersome for real life.)

Example 7 Calculating Distances

To make this a bit easier to grasp, we'll use equation 5.11 to work out the distances between some of our word vectors in Table 5.1, which contains the word vectors

$$\textit{bank} = (0, 0, 4), \quad \textit{guitar} = (1, 0, 0) \quad \text{and} \quad \textit{money} = (0, 1, 2).$$

Applying equation 5.11 gives

$$\begin{aligned} d(\textit{bank}, \textit{money}) &= \sqrt{(0-0)^2 + (0-1)^2 + (4-2)^2} \\ &= \sqrt{0+1+4} \\ &= \sqrt{5} \\ &\approx 2.24. \end{aligned}$$

In the same way, we get

$$d(\textit{bank}, \textit{guitar}) = \sqrt{1+16} = \sqrt{17} \approx 4.12.$$

Comparing these two results, we find that *guitar* is 'further' from *bank* than *money* is — a reasonable enough deduction.

However, there are problems with this technique — large vectors tend to be more distant from most other vectors than small vectors. For example, yet more calculations using equation 5.11 tell us that

$$d(\textit{guitar}, \textit{bass}) \approx 4.12 \quad \text{whereas} \quad d(\textit{guitar}, \textit{fishermen}) \approx 3.16.$$

You can confirm this by looking again at Figure 5.9, where *guitar* is slightly closer to *fishermen* than to *bass*. But this isn't really because *guitar* and *fishermen* have more in common than *guitar* and *bass* — a brief glance at Table 5.1 shows that *guitar* and *bass* have some coordinates in common whereas *guitar* and *fishermen* have none. Instead, the smaller distance between *guitar* and *fishermen* is simply because they're both nearer to the origin point. You can almost think of this as word vectors getting thrown out from a 'lexical supernova.' Each time a word is mentioned in the document collection it gets pushed futher from the origin point. The frequently occuring words get thrown the furthest and end up isolated in deep space, while the infrequent words don't get thrown very far and end up clustered around the origin, closer to one another.

Using subscripts for coordinates and the summation sign Σ takes a bit of mental gymnastics but it's well worthwhile because you can write down much more general formulas and equations without any extra effort. In particular, equation (5.11) can be used to define a distance function for a vector space of *any* dimension, without being changed at all. So far in this section, we've presumed that we're working in $\mathbb{R}^3$, where each vector a has 3 coordinates (a_1, a_2, a_3). But equation (5.11) works in just the same way for longer vectors $(a_1, a_2, \ldots, a_n)$, where the number of coordinates n can be as big as you like. In this way, Pythagoras' theorem can be adapted to give a distance measure on the vector space $\mathbb{R}^n$ for any dimension n. This measure is called the *Euclidean distance* measure on $\mathbb{R}^n$, and since it obeys the three metric space axioms (Definition 6), it is sometimes called the *Euclidean metric*.

This is another example of the fairly relaxed mentality you need to accept the ways vectors and dimensions are used. The analogy by which Pythagoras' theorem is extended from 2 dimensions to 3 dimensions should be pretty clear: the next step of applying the same numerical formulas to more than 3 dimensions isn't complicated. However, the conceptual step of trying to imagine the Euclidean distance function in $\mathbb{R}^4$ as somehow measuring the length of the diagonal of a 4 dimensional box is challenging, to put it mildly, if not downright crazy. But you don't need to do this to understand word vectors, just as we don't really need the diagram in Figure 5.9 in order to understand the significance of the

word vectors Table 5.1. The main point behind Descartes' contribution to this sort of mathematics is that *we don't need diagrams for everything* — we can work out the distances algebraically, straight from the numerical coordinates. This is the reason why these techniques can be so easily adapted to spaces with more than three dimensions which we can't visualize so easily.

5.5.4 Norms and unit vectors

We now know how to calculate the Euclidean distance and the scalar product between two vectors a and b. However, we've also seen that neither of these measures is an ideal way to work out similarities and distances between word vectors: with the Euclidean distance, frequently occurring words with large word vectors end up *too far* from most other words, and with the scalar product, the same frequently occurring words end up *too similar* to most other words. What we need is a way of factoring out these unfair advantages and disadvantages, just as we wanted to even out our graph similarity scores in Section 4.3.1.

Choosing a norm

Choosing the right norm function for a given vector space will depend on your scientific purpose in building the space. For currency vectors, the only sensible way to find the value of a whole currency vector would be to use exchange rates. At the time of writing, £1 = \$1.66, €1 = \$1.18, and ¥1 = \$0.0092, and so the vector $Dm = (6, 20, 15, 100)$ has the total value or norm

$$6 + 20 \times 1.66 + 15 \times 1.18 + 100 \times 0.0092 = \$57.82,$$

measured in dollars. In effect, we are using a 'weighted Manhattan metric' (page 102) to evaluate the length of a currency vector. (In practice, the norm of a currency vector should allow negative as well as positive values, in case you have more debts than assets — not all financial distances are positive.)

To do this, we measure the size or length of each vector, which is its distance from the zero point or origin. Applying equation (5.11) to the vectors $0 = (0, 0, \ldots, 0)$ and $a = (a_1, a_2, \ldots, a_n)$, we have

$$d(0, a) = \sqrt{\sum (a_i)^2}.$$

This can also be expressed in terms of the scalar product, since $\sum (a_i)^2$ is the same as $\sum a_i a_i$ which is just $a \cdot a$. This is summed up in the following definition.

Definition 14 The *norm* or *length* of the vector a is written $||a||$ and is defined to be

$$||a|| = \sqrt{\sum (a_i^2)} = \sqrt{a \cdot a}.$$

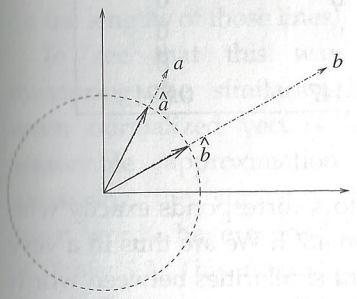


FIGURE 5.11 Normalized or 'unit' vectors

Just as the norm $||a||$ can be defined as the Euclidean distance $d(0, a)$, so also the Euclidean distance between two vectors a and b can be obtained by the formula $d(a, b) = ||b - a||$, which is the norm of the journey vector from a to b. So in many ways, whether we choose to talk in terms of norms or distances is a matter of convenience, just as whether we think of a vector as a point or as a journey from the origin to that point is a matter of convenience.

This norm is exactly the factor we need to divide by in order to remove any extra preference or penalty given to the word vectors of frequently occurring words. It's easy to check that if you divide every coordinate in a by the norm $||a||$, you are left with a vector whose norm or length is equal to 1. This vector is written as $\frac{a}{||a||}$ or sometimes $\hat{a}$, and is called the *unit vector* of a. In other words, $\hat{a}$ is the vector in the same direction as a whose length is just one unit. One simple way to think of the unit vector $\hat{a}$ is as the projection of the vector a onto a sphere or circle of radius 1, since all the points on this sphere or circle have a distance of one unit from the origin (Figure 5.11). Since all unit vectors have the same length, the only thing that distinguishes these vectors is their directions.

We can now replace all the word vectors in Table 5.1 with their unit vectors, giving the normalized version in Table 5.2. What we've done is found the norm of each row and divided each entry in that row by this number. To check that these normalized vectors *are* unit vectors, simply go along each row in the table, square each individual number and add together the results — you should get the answer 1 (or at least, as nearly as our approximation to 3 decimal places will allow).

TABLE 5.2 The vectors from Table 5.1 after they have been normalized

	Document 1	Document 2	Document 3
bank	0	0	1
bass	0.447	0.894	0
commercial	0	0.707	0.707
cream	1	0	0
guitar	1	0	0
fishermen	0	1	0
money	0	0.447	0.894

5.5.5 Cosine similarity in $\mathbb{R}^n$

The scalar product of two *normalized* vectors corresponds exactly with their *cosine similarity*, as defined in Section 4.2.1. We are thus in a very useful practical situation: we can work out similarities between words simply by working out the cosine similarities between their vectors in Table 5.2 — by multiplying together the corresponding coordinates and adding the results. For example, we now have

$$\cos(\textit{guitar}, \textit{cream}) = 1 + 0 + 0 = 1,$$

$$\cos(\textit{guitar}, \textit{bass}) = 0.477 + 0 + 0 = 0.477,$$

and

$$\cos(\textit{guitar}, \textit{fishermen}) = 0 + 0 + 0 = 0.$$

Now we have that *guitar* and *cream* are the most similar pair (which with the meaning of *cream* in Document 1 is what we should have), that *guitar* and *bass* have a certain amount in common (in fact, they have one of the meanings of *bass* in common but not both), and *guitar* and *fishermen* are completely unrelated. The skewing of such results because of *bass* being a longer vector, and recurrent problems of this nature, are gone for good.

The cosine similarity of any pair of vectors (not just unit vectors) can easily be obtained by taking their scalar product first and then dividing by their norms (rather than normalizing all vectors first and then computing cosine similarities between these normalized vectors). This is probably the most usual way of defining cosine similarity, and you will often see equations like

$$\cos(a, b) = \frac{a \cdot b}{||a||\ ||b||} \tag{5.12}$$

used in the literature to define the cosine similarity between two vectors a and b. As we said in Section 4.2.1, this number can also be thought of as the cosine of the angle between the vectors a and b, which is a good way to think of this measure of similarity which makes it intuitively clear that we are only interested in the directions, not the lengths of the vectors (since the angle between two lines doesn't depend in any way on the lengths of those lines).

To see that this way of measuring the similarity between normalized vectors is a reasonable approximation to measuring the similarity between words, have a good look at Table 5.3. This finally gives the similarity between *each* pair of word vectors. Remember that this is all calculated from the fragment of language contained in Documents 1, 2 and 3 in Figure 5.8, and in this tiny sample many of the words only occur with a fraction of their possible meanings. Given this, Table 5.3 does a remarkably good job of modelling which words are similar and dissimilar to one another.

Such a table of similarities between each pair of objects in a collection can be called a *data matrix* or an *adjacency matrix*. Note that the main top-left to bottom-right diagonal is made entirely of 1's — this is because each word has a similarity of 1 with itself, because the length of each normalized vector is equal to 1. If we reflect the table about this diagonal (thereby interchanging the rows and the columns), notice that we get the *same* table. A table (or matrix) with this property is called *symmetric*,

Cosine similarity and Euclidean distance

The cosine similarity and Euclidean distance between two unit vectors are closely related: if a and b are unit vectors then it follows that

$$\begin{aligned}(d(a,b))^2 &= ||b-a||^2 \\ &= (b-a)\cdot(b-a) \\ &= b\cdot b + a\cdot a - 2a\cdot b \\ &= 2 - 2a\cdot b.\end{aligned}$$

It follows that the relative ranking of points as being 'more or less similar' according to cosine similarity or 'closer or further away' according to Euclidean distance will be the same for unit vectors.

However, because the left hand side in this equation is squared, cosine similarity is 'less transitive' than Euclidean distance. For example, the vectors $a = (1,0)$, $b = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ and $c = (0,1)$ all have length 1, and although $\cos(a,b) = \cos(b,c) = \frac{\sqrt{2}}{2} \approx 0.707$ which is quite high, $\cos(a,c)$ is still 0.

TABLE 5.3 The cosine similarities between each pair of words in our model

	bank	bass	commercial	cream	guitar	fishermen	money
bank	1	0	0.707	0	0	0	0.894
bass	0	1	0.632	0.447	0.447	0.894	0.400
commercial	0.707	0.632	1	0	0	0.707	0.915
cream	0	0.447	0	1	1	0	0
guitar	0	0.447	0	1	1	0	0
fishermen	0	0.894	0.707	0	0	1	0.447
money	0.894	0.400	0.915	0	0	0.447	1

because it is symmetrical in the traditional "it looks the same if you reflect it in a mirror" fashion, and because of this geometric symmetry, these matrices can be used to represent symmetric *relations* (page 60).

5.6 Document vectors and information retrieval

This chapter was meant to tell you how a search engine works, and so far it's been entirely about *words* and these things called vectors, whereas one thing that all readers will probably know is that search engines are about finding *documents*. In case you're feeling tricked into reading a whole chapter on mathematics under false pretences, we will finish this chapter by describing how to create *document vectors* so that we finally have a little system which can take a query made out of any combination of the seven terms in Table 5.3 and rank the three documents in Figure 5.8 according to their relevance to such a query.

Norms and Euclidean distances of vectors are just as easy to adapt to n dimensions: in some ways, computers find this easier than people because computers don't care whether or not they can visualize the vectors they're using.

The trick is very simple: we represent the documents as vectors in *the same space* as the words, and then we can compute similarities between words and documents just as we computed similarities between pairs of words. In a sense, we've already been doing this — our word vectors have had three coordinates, one for their occurrence in each of the documents, and in this sense the three documents have been used as

a *basis* for our space of word vectors. One way to make this clearer is to reason that if we had 4 documents in our collection, we would need 4 columns in the term-document matrix and each word vector would have 4 coordinates, and so on for n documents: so the number of documents clearly determines the dimension of the space of word vectors. Another way to see this pictorially is to consult the diagram in Figure 5.9, where the 3 documents are clearly being used as 3 coordinate axes. Representing each document as a unit vector in the direction of its coordinate axis, we have the *document vectors*

$$\textit{Doc1} = (1,0,0), \quad \textit{Doc2} = (0,1,0) \quad \text{and} \quad \textit{Doc3} = (0,0,1).$$

It's now a simple matter to work out the cosine similarity between a given query expression and each document in turn. For example, for the query *bass* with vector $(0.477, 0.894, 0)$, we get

$$\cos(\textit{bass}, \textit{Doc1}) = 0.477, \qquad \cos(\textit{bass}, \textit{Doc2}) = 0.894,$$
$$\text{and} \qquad \cos(\textit{bass}, \textit{Doc3}) = 0.$$

If the user was returned the winning document *Doc2* and realized that this isn't the sort of *bass* they were looking for, they could add the term *guitar* to give the query *bass guitar*. Now the benefits of using vectors really begin to pay off — we can simply add together the vectors for *bass* and *guitar* to give a new *query vector*,

$$\textit{bass} + \textit{guitar} = (0.477, 0.894, 0) + (1, 0, 0) = (1.477, 0.894, 0),$$

which when normalized becomes $(0.855, 0.518, 0)$.

Comparing each document with this new query vector, *Doc1* is the winner with a cosine score of 0.855. In a very simple way, we have combined the meanings of two words to give a meaning for a combination of words, for the first time in this book. This very important process is called *semantic composition*, which we are modelling here (very bluntly) using vector addition.

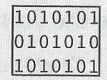
You can carry on playing the game of working out query vectors, comparing them with the three document vectors, and see if the ranking you get coincides with how relevant the documents are to your query.

That is one basic way to build an information retrieval system, though there are many important questions to ponder. Many researchers and engineers have improved over this baseline, finding out how to assess the importance of different terms and documents (Google's *PageRank* is

one example of an answer to such a question), how the vector model compares with other conceptual models, how different terms might depend on one another, and how to cope with the engineering task of managing bigger and bigger document collections and term-document matrices. The wider reading section at the end of this chapter will barely be able to scratch the surface of these topics, though it will try to give some good leads which you can follow up for yourself.

This may seem like a lot of mathematics in order to end up multiplying a few numbers here and there and coming up with a ranking of our three documents. In many ways, this is one of the vector model's great strengths — the mathematics in this chapter may have seemed to have some strange names and symbols and far too many dimensions, but at the end of the day this is nothing more than a shorthand for adding and multiplying lots of different numbers. These numbers can be extracted straight from a document collection without any need for deep linguistic analysis, which is one of the main reasons that information retrieval has proved to be such a widespread and successful branch of natural language processing: your system really doesn't need to know much about *language* at all. (If you just stop for a moment to think about how much more trouble we'd have had in building a simple system that could read out a spoken version of three documents or translate them into a different language getting both the grammar and the meaning correct, I think you'll agree that building an information retrieval system was pretty easy.)

Another vital point is that the toy system we've described in this chapter is *scalable*. This is where the mental gymnastics of working in many dimensions pays off. It may have seemed like unnecessarily hard



Programming cosine similarity in higher dimensions

If we use arrays to hold our vectors and a `for` loop to work out the cosine, cosine similarity in n dimensions can be implemented using exactly the same code as for 2 dimensions. If your (normalized) vectors are called `vec1` and `vec2` and their cosine similarity is stored by the variable `cos`, the code would read something like

```
cos=0;
for( i=0; i < dim; i++){
 cos=cos+vec1[i]*vec2[i];
}
```


work to develop all this n dimensional mathematics when most of the time we were just working in 2, 3 and 4 dimensions. But towards the end of the chapter we realized that we were using a dimension for each separate document. Now, if we'd confined ourselves to a comfortable mathematical system that could just cope with 2 or 3 dimensions, we'd have no hope of coping with a bigger document collection. But instead, we put in the hard work to use a mathematical language that can be adapted to *any* number of dimensions: so the same system can be used for a document collection of any size.[8]

I'd like to finish this chapter with two observations which take us back to the concepts of Chapter 1. We said in Section 1.5.4 that the 'measurements' we made of the extent to which different words occurred in different relationships or contexts would be *discrete*, but that many of the models we would build from this information would rely on *continuous* mathematical techniques. The vector spaces of this chapter are precisely the sort of continuous model to which we were referring. We started by counting discrete whole numbers (for example, in Table 5.1), but by the time we had normalized these vectors in Table 5.2, we needed all sorts of numbers in between 0 and 1, and in principle the only limit on the specificity of these numbers is how many decimal places it's worthwhile keeping for each of them. When we're just comparing the relevance of three different documents, it doesn't matter so much: when we're dealing with a document collection the size of the World Wide Web, we may need more and more in-between numbers to keep track of ever finer distinctions of importance and relevance. The binary partition into relevant and non-relevant documents could not support a user who needed to know which out of thousands of relevant documents were the *most* relevant, and this challenge has naturally led researchers to

[8] In the presentation in this chapter, I have departed slightly from the approach in most textbooks, which uses the words or *terms* as coordinate axes from which to represent both document and query vectors. Instead I have used the documents as coordinate axes for repreenting the terms, because this forms a better introduction to the way we will use word vectors in the rest of the book, where we are much more interested in computing word-word similarities than (say) document-document similarities. Also, it was much easier to try to draw Figure 5.9 as 7 word points in 3 document dimensions than as 3 document points in 7 word dimensions! Some mathematical issues related to such questions are explored by Yang et al. (1998) in the context of translingual information retrieval.

study continuous methods such as vector spaces, fuzzy set theory and probability.

Continuous methods enable us to model not only atoms of meaning such as words, but the space or void in between these words. Whole passages of text are mapped to points of their own in this void, without changing the underlying shape of the space around them, and we can measure the distance between query terms and these document points very easily, if naïvely.

Our models have come all the way from Chapter 2, where we studied relationships between words almost without using any numbers at all, to the work in this chapter where words have been represented using *only* numbers, obtained by measuring how many times a word occurs in different documents. Geometric comparisons such as longer or shorter, closer or further away, have been made purely by doing calculations with these coordinates. This has all been made possible by the 'Cartesian revolution' in mathematics which enabled geometric problems to be described and solved using relationships between collections of numbers.

Wider Reading

The introduction to vectors given here is necessarily brief and very informal. Those who wish to go deeper or more thoroughly into any of the concepts introduced here should consult a proper book on linear algebra. Chapters 2 and 3 of Jänich (1994) are appropriate and readable: another good introduction is Vallejo (1993).

The rest of this book will describe many important operations with vectors, and the ways they have been used for analyzing linguistic information: a lengthy list of wider reading on this topic will gradually emerge. Those interested in the particular uses of vectors for information retrieval should consult the classic text of Salton and McGill (1983, Ch 4), and the summary contained in Jurafsky and Martin (2000, §17.3) which gives a good pictorial overview.

The principal mathematical models used for information retrieval systems are the *vector*, *probabilistic* and *fuzzy set* models, all of which are discussed in Baeza-Yates and Ribiero-Neto (1999). The fuzzy set model is an adaptation of the traditional Boolean model (which will be described in Section 7.1) to cope with continuous values. Fuzzy sets

for information retrieval are discussed at length by Miyamoto (1990), starting with a good introduction to what fuzzy sets are, and providing fuzzy set models for some of the traditional thesaural relationships such as 'Broader Term' which we met in Chapter 2. One of the most interesting probabilistic models is the 'inference network' used in the INQUERY system (Turtle and Croft, 1989, Callan et al., 1992): such networks use probability theory to find important paths in directed graphs of the sort we studied in Chapter 3, in this case the path that leads between a query and relevant documents.

A unifying theme behind these three models is that normalized vectors, probability distributions and fuzzy sets are all mathematical techniques which replace binary values (0 or 1) which signify 'not belonging' or 'belonging' with continuous values (0 or 1 or anything in between) which signify 'partially belonging' or 'probably belonging.' There are some practical differences between the models (for probability distributions, the sum of all the individual probabilities must be equal to 1, whereas as we've seen for normalized vectors, the sum of the *squares* of the individual coordinates must be equal to one, at least with the Euclidean norm), but it seems at least possible that these are really different *implementations* of one underlying *model*.

Other important topics in information retrieval include weighting strategies, user models and interfaces, linguistic operations such as stemming words (such as *fishermen*) to their lexical roots (such as *fisherman* or even *fish*) and of course engineering. Rather than try to give piecemeal and unsatisfactory pointers to each of these topics, I recommend consulting the indexes of general books such as Salton and McGill (1983), Kowalski (1997) and Baeza-Yates and Ribiero-Neto (1999), and also the excellent range of papers collected by Sparck Jones and Willett (1997).

Those interested in the mathematical development of vectors should certainly look at the pioneering work of Descartes (1637). The use of numbers and lines together to solve problems is pioneered in Book I, which in particular introduces the idea of describing points by measuring their coordinates on two chosen lines (p. 29, p. 310 in the original French). There is a particularly beautiful version available from Dover which contains a facsimile of Descartes' original French manuscript next to the English translation. An excerpt from the vital first

book of the Analytic Geometry is contained in Smith (1929, p. 397-403), and a good historical account can be found in Boyer and Merzbach (1991, Ch. 17).

The technique of using vectors to represent points in space grew from the work of Sir William Rowan Hamilton (1805-1865) on quaternions, in which he first describes the addition of a fourth dimension to a mathematical system (Hamilton, 1847). At the same time, during the 1840s, a little known German high school teacher called Hermann Grassmann (1809-1877) was developing a theory of *Ausdehnungslehre* (*extension theory*), and this work contains our modern notion of vectors in *any* number of dimensions (Grassmann, 1862). Since his purpose was to represent a general concept with any number of dimensions, our word vectors owe much more to this foundation — and as we shall see in Chapter 7, Grassmann contributed some of the tools which are key to adapting the power of logic to the geometric setting of a vector space.

In cognitive science, the use of coordinates and dimensions to model mental processes is the cornerstone of the *Conceptual Spaces* of Gärdenfors (2000). In particular, the first chapter of this book contains an excellent introduction to the notion of dimensions, and the way different stimuli such as taste and colour can be represented as points in such a conceptual space. For example, colours seem to have a persistent 3 dimensionality about them, whether those dimensions are 'red, green and blue' or 'hue, brightness and chromaticity.'

No bibliography about dimensions would be complete without mentioning Edwin Abbott's novel *Flatland*, originally published in 1884. This slim book (under 100 pages) is the darling of every enthusiast who's ever read it and the Dover edition will probably cost less than your bus fare to the bookshop to buy it. In Abbott's story (also a barbed satire against the hierarchical rigidity of Victorian society) the world of a humble square is rocked by the intersection of a solid sphere with his two-dimensional existence. At first resistant, the square becomes convinced that a *third dimension* is possible, and through trying to explain this to the other polygons he becomes a prophet, a heretic and finally a prisoner, clinging forlornly to the hope that his teachings will one day inspire a new generation "who shall refuse to be confined to limited Dimensionality."

6

Exploring Vector Spaces in One and More Languages

This chapter is *much* easier than the last one. Having gone through the mathematics and (in a small example) the process of building word vectors, this chapter concentrates entirely on the reward for these achievements: being able to learn a lot about words themselves by exploring their vector spaces. The tour guide nature of this book is being given free-rein: you've put in (or adroitly skipped!) the hard mathematical labour to get to this point, and now you get to be a tourist and enjoy the view.

Not that this chapter is without new ideas. We'll talk about different ways of making the views more informative than a simple list of related words, by grouping words into clusters or plotting their vectors in a two-dimensional 'word-spectrum.' We'll also see how to use documents from two languages to build a single vector space with words from *both* languages, which can (for example) be used to translate words and queries between languages.

While this hopefully makes a cohesive and readable story, you don't have to read it in sequence, and a perfectly acceptable way to approach this chapter is to flick through, see which of the pictures interest you, and delve into those sections to see how they were made. Or maybe even better, follow the links that I've highlighted at the beginning of sections and explore the models online for yourself. They say that "seeing is believing," and while the examples chosen for this chapter work especially well at highlighting particular points, there's no better way to convince yourself that the techniques we've described really do